Универзитет у Београду

Математички факултет

Фибоначијев хип и примене

Мастер рад

име и презиме	**Мила Ратковић**
број индекса	1010/2012
школска година	
професор	др Миодраг Живковић
датум	

Садржај

Глава 1

Увод

Живимо у тренутку када су информације јако вредне. Свакодневно се прикупљају терабајти података о томе како крајњи корисници користе разне производе. У складу са растом количине прикупљених података расла је и потреба да се то мноштво података ефикасно анализира како би се издвојили подаци који могу да се употребе. Основа сваке анализе је претрага. Са временом, смишљене су разне структуре података како би се претрага учинила бржом и ефикаснијом. Једна од ефикаснијих структура за претрагу великог скупа података је Фибоначијев хип. У наставку рада је приказан теоријски концепт структуре Фибоначијев хип. Објашњене су основне операције са том структуром, како се користи у теорији и пракси, као и програмска реализација саме структуре. На крају је серијом експеримената утврђен однос између перформанси бинарног хипа и Фибоначијевог хипа. С обзиром на то да се Фибоначијев хип не користи толико често у пракси, постоји јако мало литературе која обрађује ову структуру података. У раду је највише коришћена књига ”Intoduction to Algorithms" која је наведена у делу о литератури.

Глава 2

Хип

2.1 Бинарни хип

Хип је бинарно стабло које задовољава *услов хипа*: кључ сваког чвора је мањи или једнак од кључева његових синова. Непосредна последица дефиниције (због транзитивности релације $\leq$) је да је у хипу сваки чвор мањи или једнак од кључева његових потомака.

Хип је погодан за реализацију *листе са приоритетом*, апстрактне структуре података за коју су дефинисане две операције:

- **umetni**(x) - уметни у структуру кључ x
- **ukloni**() - обриши најмањи кључ из структуре.

Бинарни хип се може реализовати имлицитно или експлицитно задатим стаблом. У наставку се користи имплицитна реализација јер се операције могу изводити тако да стабло увек буде уравнотежено.

Дакле, кључеви хипа су смештени у вектор A, при чему је k горња граница за број елемената хипа. Ако је n текући број елемената хипа онда се за смештање елемената хипа користе локације у вектору од 1 до n. Притом, ако је елемент на i-тој позицији, онда су његови синови на позицијама $2i$ и $2i + 1$.

Под висином h стабла се подразумева број чворова од корена до најудаљенијег листа.

Комплетно бинарно стабло висине h је стабло са n чворова у коме би се тих n чворова могло нумерисати бројевима од 1 до n, као када би то било првих n чворова у пуном стаблу висине h. Пуно бинарно стабло је бинарно стабло висине h које садржи $2^k - 1$ чворова.

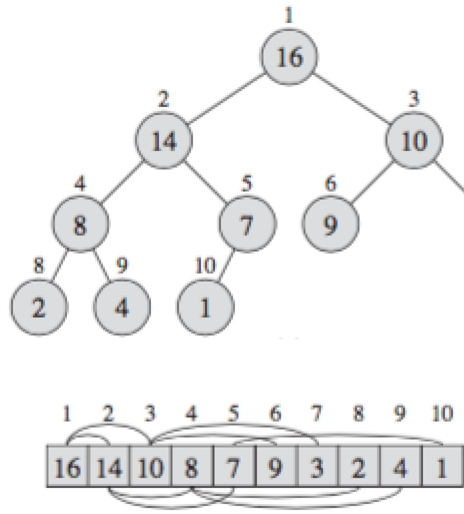
Комплетно стабло се попуњава с лева на десно:

- сви листови су на истом нивоу или на два суседна нивоа,
- сви чворови на најнижем нивоу се налазе што је могуће више лево.

Уклањање са хипа елемента са најмањим кључем. Кључ најмањег елемента је $A[1]$, па га је лако пронаћи. После тога је потребно трансформисати преостали садржај вектора тако да представља исправан хип. То се може постићи тако што се $A[n]$ прекопира у корен $A[1]$ и број елемената n, смањи за један. Означимо са $x = A[1]$, нови кључ корена, а са $y{=}min\{A[2]{,}A[3]\}$ мањи од кључева синова корена. Подстабла са коренима у $A[2]$ и $A[3]$ су исправни хипови, ако је $x \leq y$ онда је комплетно стабло исправан хип. У противном ако је $x \geq y$, после замена места кључева x и y, у стаблу, односно вектору A, подстабло са непромењеним кореном остаје исправан хип, а у другом подстаблу је промењен (увећан) кључ у корену. На то друго подстабло се примењује рекурзивно (индукција) иста процедура која је примењена на полазно стабло. Индуктивна хипотеза је да ако је после i корака x на позицији $A[j]$, онда само подстабло са кореном $A[j]$ евентуално не задовољава услов хипа. Даље се x упоређује са $A[2j]$ и $A[2j+1]$ (ако постоје) и ако није мањи или једнак од оба ова кључа, замењује место са већим од њих. Дакле, кључ x се спушта на доле, све док не доспе у лист или у корен неког подстабла у коме је мањи или једнак од кључа оба сина.

Уметање новог елемента у хип. Број елемената n се повећа за један и на слободну позицију $A[n]$ се упише нови кључ. Тај кључ се упоређује са кључем оца $[i]$, где је $i{=}\lceil n/2 \rceil$, па ако је мањи од њега, замењују им се места. Ова замена може само да умањи кључ $A[i]$, па ће чвор са кључем $A[i]$ бити корен исправног хипа. Може се показати да је следећа индуктивна хипотеза тачна: ако је уметнути кључ после низа премештања доспео у елемент $A[j]$, онда је стабло са кореном $A[j]$ исправан хип, а ако се подстабло уклони, остатак стабла задовољава услов хипа. Процес се наставља премештањем кључа навише, све док не буде већи или једнак од кључа оца, или док не дође до корена, тада комплетно стабло задовољава услов хипа.

Број упоређивања приликом извршавања обе операције је ограничен висином стабла, јер је потребно извршити највише h замена корена подстабла са једним од његових потомака како би у потпуности важило својство хипа, односно $O(\log n)$. Недостатак бинарног хипа је у томе што се тражење задатог кључа не извршава ефикасно. Осим тога, спајањем два низа који репрезентују бинарне хипове коришћењем операције креирања минималног хипа, операција **unija**, се у најгорем случају извршава за време $\Theta(n)$. Операција креирања минималног хипа се извршава тако што се креће кроз низ од елемената који је отац чвора i до 1. елемента низа. Позиција чворова синова и очева се мења ако нису у исправном поретку.



Слика 2.1: Пример бинарног хипа представљеног преко стабла, односно низом. Кључ сваког чвора већи је од кључева синова.

2.2 М-хип

М-хип је апстрактна структура података која подржава следеће операције: **kreiraj**, **ubaci**(H, x), **minimum**(H), **izvadi_min**(H), **unija**(H_1, H_2). Операције као што су: **ubaci**(H, x), **unija** (H_1, H_2), **umanji_ključ**(H, k, x) Фибоначијевог хипа се извршавају за константно амортизовано време, за разлику од истих операција бинарног хипа. Због тога је ова структура јако погодна за апликације које учестало позивају претходне три операције.

М-хип је структура података у којој сваки елемент (чвор) има кључ и која подржава следећих пет операција.

1. **kreiraj** () - креира и враћа нови хип који не садржи ниједан елемент.
2. **ubaci**(H, x) - убацује елемент x коме је придружен кључ у хип H.
3. **minimum**(H) - враћа показивач на елемент у хипу H чији је кључ најмањи.
4. **izvadi_min**(H) - брише из хипа H елемент са најмањим кључем и враћа показивач на елемент.
5. **unija**(H_1, H_2) - креира и враћа нови хип који садржи све елементе хипова H_1 и H_2. Овом операцијом се бришу хипови H_1 и H_2.

Фибоначијев хип (тачка 2) је реализација М-хипа која осим стандардних операција М-хипа, подржава још две операције:

1. **umanji_ključ**(H, x, k) - додељује елементу x из хипа H нову вредност кључа, за коју се претпоставља да није већа од тренутне вредности кључа.

2. **ukloni**(H, x) - уклања елемент x из хипа H.

Подразумевана варијанта М-хипа подржава операције **minimum**, **izvadi_min**, **umanji_ključ**. Аналогно се може користити симетрична варијанта М-хипа са операцијама **maksimum** и **uvećaj_ključ**.

У табели 2.1 приказано је поређење сложености операција бинарног и Фибоначијевог хипа (погледати поглавље 3). За бинарни хип су приказане сложености операција у најгорем случају, док су за Фибоначијев хип приказне амортизоване сложености, односно просечне сложености за дугачак низ операција истог типа. У тачки 3, приказани су резултати поређења бинарног у односу на Фибоначијев хип у операцијама **ubaci**, **minimum**, **izvadi_min** и **umanji_ključ**.

Операције	Бинарни хип (најгори случај)	Фибоначијев хип (амортизовано време)
kreiraj_hip	$\Theta(1)$	$\Theta(1)$
ubaci	$\Theta(\log n)$	$O(1)$
minimum	$\Theta(1)$	$\Theta(1)$
izvadi_min	$\Theta(\log n)$	$\Theta(\log n)$
unija	$\Theta(n)$	$\Theta(1)$
umanji_ključ	$\Theta(\log n)$	$\Theta(1)$
ukloni	$\Theta(\log n)$	$\Theta(\log n)$

Табела 2.1: Упоређивање сложености операција бинарног хипа и Фибоначијевог хипа

У табели 2.1, Фибоначијев хип има бољу асимптотску временску границу него бинарни хип за операције **ubaci**, **unija**, **umanji_ključ**, док је сложеност осталих операција иста као код бинарног хипа. Уколико у апликацији није потребна операција **unija** бинарни хип је сасвим прихватљиво решење. Међутим, ако је она неопходна, перформансе бинарног хипа су знатно лошије.

Глава 3

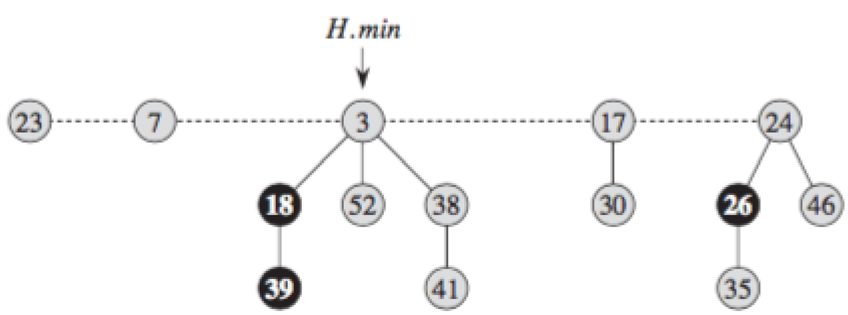
Структура Фибоначијевог хипа

Свака структура података се састоји од чворова. Чворови се различито дефинишу у зависности од структуре података која их садржи.

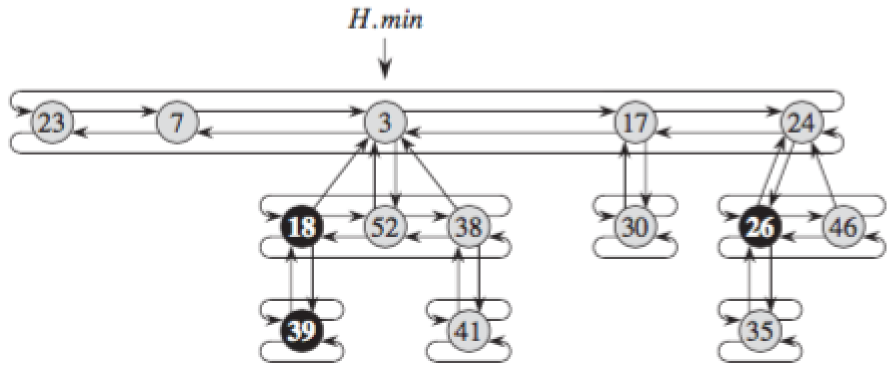
Чвор стабла се састоји од кључа (вредности) и показивача који садрже адресу сина.

Чвор двоструко повезане листе се састоји од кључа (вредности) и показивача који садрже адресу претходног и следећег чвора.

Фибоначијев хип (Ф-хип) је кружна листа коренова стабала која задовољавају својство (услов) хипа, односно сваки чвор стабла има вредност кључа већу или једнаку од вредности кључа свог оца.



Слика 3.1:
Пример Фибоначијевог хипа.



Слика 3.2:
Детаљна структура Фибоначијевог хипа.

Слика илуструје како су повезани чворови Фибоначијевог хипа. Као што видимо, отац чвора 3 је NIL, док је чвор са кључем 52 један од синова. Листа деце чвора 3 је састављена од чворова 18, 52 и 38. Лево од чвора 52 је чвор 18, десно од чвора 52 је чвор 38. Чвор 39, 41, 30 или пак 35 су једини синови свог оца. Прецизнији опис структуре Ф-хипа следи у наставку.

Сваки чвор x садржи показиваче $x.otac$ који показује на оца и $x.sin$ који показује на сина чвора x. Синови чвора x су кружно повезани, двоструко повезаном листом, коју називамо *листом синова чвора* x. Сваки син y у листи синова, има показиваче $y.levo$ и $y.desno$ који показују на леви и десни потомак, респективно.

Потомци се могу појавити у било ком поретку у листи синова. Чињеница да се ова структура састоји из кружних, двоструко повезаних листа, има одређених предности. Може се додати и обрисати чвор са било које локације за време O(1), а могуће је и спајање две кружно повезане листе у једну, за време извршавања O(1).

Уколико је y једини син, онда је задовољен следећи услов:

$$y.levo = y.desno = y$$

Сваки чвор x има следеће атрибуте:

- $x.stepen$ - садржи број синова чвора x,

- $x.markiran$ - приказује да ли је обрисан неки од синова чвора x после тренутка када је x постао син неког новог чвора. Атрибут може да има вредност true или false. Чворови који су маркирани на сликама су означени црном бојом.

Ново-креирани чворови су немаркирани и сваки пут када x постане дете неког новог чвора, његов атрибут $x.markran$ добије вредност $false$. Ово се дешава у оквиру операције *upis* о којој ће касније бити речи.

Фибоначијевом хипу H се приступа преко показивача $H.min$ који показује на чвор корена стабла са најмањом вредношћу кључа. Овај чвор се зове ***минимални чвор*** Фибоначијевог хипа. Уколико Фибоначијев хип има више чворова чији кључеви имају исту вредност која је притом и минимална у односу на цео хип, онда било који од тих чворова може бити минимални чвор $H.min$. Када је Фибоначијев хип празан онда је задовољен услов $H.min = NIL$.

Атрибут Ф-хипа који представља укупан број чворова се обележава са $H.n$.

Сваки чвор стабла у Фибоначијевом хипу је повезан преко левог и десног показивача кружном, двоструко повезаном листом, са суседним коренима, али исто тако је повезан и са својим сином и оцем. Листа која је састављена од двоструко повезане листе суседних чворова, који немају оца, се назива ***корена листа*** Фибоначијевог хипа. У оквиру Фибоначијевог хипа, стабла са могу појављивати у било ком поретку.

3.1 Метода потенцијала за анализу амортизоване сложености

За анализу перформанси операција Фибоначијевог хипа користи се метода ***потенцијал***.

Метода потенцијала ради на следећи начин:

извршава се n операција, почињући од иницијалне структуре D_0. За свако $i = 1, 2, 3, .., n$ нека је c_i цена i-те операције и нека је D_i структура података која је настала као резултат примене i-те операције на структуру података D_{i-1}. Функција потенција $\Phi(D_i)$ пресликава структуру података у неки број. Овде се уводи и амортизована цена i-те операције која се дефинише на следећи начин:

$$\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1})$$

Према томе, амортизована цена је једнака збиру стварне цене и промене потенцијала.

На основу претходног, укупна амортизована цена при извршавању n операција је:

$$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} (c_i + \Phi(D_i) - \Phi(D_{i-1})) = \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0)$$

Ако дефинишемо потенцијалну функцију Φ за коју важи: $\Phi(D_n) \geq$

$\Phi(D_0)$, тада добијамо горњу границу укупне цене јер важи да је:

$$\sum_{i=1}^{n} \hat{c}_i \geq \sum_{i=1}^{n} c_i$$

У пракси не знамо колико ће бити потребно операција при извршењу, тако да једноставно може да се каже да је потребно да се задовољи услов $\Phi(D_i) \geq \Phi(D_0)$ за свако i. Обично се дефинише да је $\Phi(D_0)$=0 и након тога се покаже: $\Phi(D_i) \geq 0$ за свако i.

Интуитивно, ако је разлика потенцијала i-те операције позитивна ($\hat{c}_i > c_i$), онда је амортизована цена c_i претплаћена и потенцијал структуре података се повећава. Међутим, ако је разлика негативна, онда није довољно плаћена i-та операција и потенцијал се смањује.

Стога амортизована цена зависи од избора потенцијалне функције Φ. Различите потенцијалне функције могу водити до различитих амортизованих цена, а да амортизована цена и даље остане горња граница стварне цене операције.

Сваки Фибоначијев хип H има следеће карактеристике:

- $s(H)$ - број стабала у кореној листи и
- $m(H)$ - број маркираних чворова.

Потенцијал $\Phi(H)$ Фибоначијевог хипа H се може дефинисати изразом:

$$\Phi(H) = s(H) + 2m(H)$$

Потенцијал је ненегативан број. Потенцијал празног хипа је једнак нули. Потенцијал скупа Фибоначијевих хипова је једнак суми појединачних потенцијала Фибоначијевих хипова.

Глава 4

Операције М-хипа применом Фибоначијевог хипа

За разлику од бинарног хипа, М-хип операције одлажу посао што је дуже могуће.

4.1 Максимални степен

Претпоставимо да знамо горњу границу, $D(n)$, максималног степена (број деце чвора) било ког чвора у Фибоначијевом хипу, где је n број чворова Фибоначијевог хипа. Уколико су подржане само М-хип операције онда је:

$$D(n) \leq \lfloor \log n \rfloor$$

Међутим, ако су подржане и операције **umanji_ključ** и **ukloni** онда је:

$$D(n) = O(\log n)$$

Доказ претходне тврдње је у тачки 6, последица 6.0.4.1.

Ако имамо празан Фибоначијев хип и убацујемо k чворова, Фибоначијев хип би се састојао само од листе са k чворова који представљају коренове стабала. Ако се након убацивања примени операција **izbaci_min** (видети поглавље 4.6), чвор који има минималну вредност кључа се избацује из листе и показивач $H.min$ се поставља на нови чвор са најмањим кључем. Да би се пронашао нови минимум, неопходно је да се претражи $k-1$ чворова. У току проналажења минималног чвора, чворови се могу повезати у стабло са мањим бројем чворова у кореној листи. Величина нове корене листе је највише $D(n)+1$. Главна идеја која се овде користи је да сви чворови имају различите степене (а највећи степен чвора је по претпоставци $D(n)$).

Без обзира на то како је корена листа раније изгледала, применом операције **izbaci_min**, сваки чвор из листе ће имати степен који ће се разликовати од степена било ког другог чвора из те исте листе. Величина нове корене листе након **izbaci_min** је највише $D(n)+1$.

4.2 Креирање Фибоначијевог хипа

Да би се креирао празан Фибоначијев хип, потребно је применити операцију **kreiraj** која алоцира и враћа Фибоначијев хип код кога је $H.n = 0$ и $H.min = NIL$. Пошто у њему нема стабала, јер је број чворова једнак нули, односно $s(H) = 0$, и важи да је $m(H) = 0$, онда је потенцијал празног хипа такође једнак нули, тј. $\Phi(H) = 0$. Према томе, промена потенцијала је нула, те је амортизована цена извршавања операције **kreiraj** једнака стварној цени извршавања, односно $O(1)$.

4.3 Убацивање чвора

Операција **ubaci(**H, x**)**, убацује чвор x у Фибоначијев хип H под претпоставком да чвор већ садржи $x.klju\check{c}$.

Следећем псеудокод описује алгоритам.

1. $x.stepen = 0$
2. $x.otac = NIL$
3. $x.sin = NIL$
4. $x.markiran =$ **false**
5. **if** $H.min == NIL$
6.   kreiraj korenu listu za H u kojoj se nalazi samo čvor x
7. $H.min = x$
8. **else** ubaciti x u korenu listu Fibonačijevog hipa H
9.   if $x.klju\check{c} < H.min.klju\check{c}$
10. $H.min = x$
11. $H.n = H.n + 1$

Као што се види из псеудокода, линије један, два, три и четири иницијализују атрибуте чвора x. Линија пет тестира да ли је Фибоначијев хип празан. Ако јесте, онда се чвор x убацује у корену листу која је пре операције била празна. Након тога, поставља се показивач $H.min$ на чвор x. Уколико хип није био празан, чвор се само убацује у корену листу, и ажурира се показивач $H.min$. На крају линија једанаест увећава број чворова Фибоначијевог хипа за један због додавања новог чвора.

Да би се одредило амортизовано време извршавања потребно је израчунати промену потенцијала. Нека су H и H' Фибоначијеви хипови пре и после операције, респективно.

Како важи:

$$s(H') = s(H) + 1$$

и

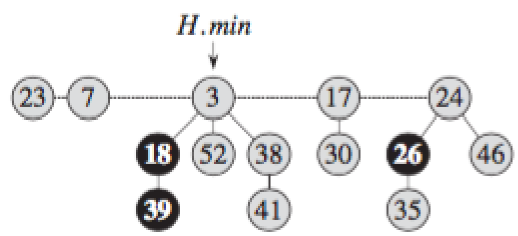
$$m(H') = m(H)$$

тада је промена потенцијала:

$$((s(H) + 1) + 2m(H)) - (s(H) + 2m(H)) = 1$$

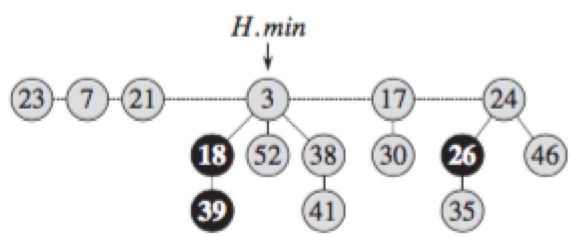
Како је стварна цена $O(1)$, амортизована цена извршавања је:

$$O(1) + 1 = O(1)$$



Слика 4.1: Фибоначијевог хипа пре операције уноса.

У овај хип убацујемо чвор 21. Он се убацује тако што се дода кореној листи. Резултат хипа је дат наредном сликом.



Слика 4.2: Фибоначијевог хипа после операције уноса

4.4 Проналажење минималног кључа

Минимални чвор у Фибоначијевом хипу се веома лако проналази. Показивач $H.min$ већ показује на чвор са минималном вредношћу кључа. На основу тога имамо да је време приступања минималном елементу једнака $O(1)$. Како се за проналажење минималног чвора не мења потенцијал, имамо да је амортизована цена извршавања $O(1)$.

4.5 Спајање два Фибоначијева хипа

Операција **unija** (H_1, H_2) спаја два Фибоначијева хипа у један, при чему се хипови H_1 и H_2 бришу. Спајање се врши веома једноставно. Спајају се њихове корене листе и потом се одређује минимални елемент.

Псеудокод алгоритма за унију два Фибоначијева хипа.

1. $H =$ **kreiraj()**
2. $H.min = H_1.min$
3. spojiti korenu listu F-hipa H_2 sa korenom listom F-hipa H
4. **if** $(H_1.min == NIL)$ or $(H_2.min \neq NIL$ and $H_2.min.klju\check{c} < H_1.min.klju\check{c})$
5. $\quad H.min = H_2.min$
6. $H.n = H_1.n + H_2.n$
7. return H

Линија 1 креира празан Фибоначијев хип. Линија два поставља минимални елемент ново-креираног Ф-хипа на минимални елемент првог хипа. У следећој линији корена листа Ф-хипа H_2 хипа постаје корена листа новог хипа. Након тога се налази услов у коме се испитује да ли је први хип празан, или ако постоји други хип да ли је његов минимални елемент мањи од минималног елемтента првог хипа, онда је минимални елемент Фибоначијевог хипа H, управо тај минимални елемент другог хипа H_2. У наредној линији, број чворова ново-креираног хипа је једнак збиру чворова хипова који улазе у његову изградњу.

Промена потенцијала је следећа:

$$\begin{aligned} &\Phi(H) - (\Phi(H_1) + \Phi(H_2)) \\ &= (s(H) + 2m(H)) - ((s(H_1) + 2m(H_1)) + (s(H_2) + 2m(H_2))) \\ &= 0 \end{aligned}$$

зато што је $s(H) = s(H_1) + s(H_2)$ и $m(H) = m(H_1) + m(H_2)$.

Пошто се потенцијал није променио, амортизована цена извршавања је једнака стварној цени извршавања, тј. $O(1)$.

4.6 Уклањање минималног елемента

Приликом извођења ове операције претпоставља се да се показивачи у оквиру листе ажурирају, а да показивачи чвора који се избацује остају непромењени.

После вађења чвора са минималним кључем посебна помоћна функција пролази кроз чворове корене листе и групише их у хипове са различитим степеновима, што обезбеђује смањење дужине корене листе.

У наставку је псеудокод основног алгоритма који од чворова деце минималног чвора креира корену листу, уклањајући минимални елемент. Након тога извршава се помоћне функције која се зове сједињавање.

1. $z = H.min$
2. **if** $z! = NIL$
3.   za svakog sina x čvora z
4.   dodaj čvor x korenoj listi Fibonačijevog hipa H
5. $x.otac = NIL$
6.   ukloniti z iz korene liste hipa H
7.   **if** $z == z.desno$
8. $H.min = NIL$
9.   **else**
10. $H.min = z.desno$
11.   **sjedinjavanje**(H)
12. $H.n = H.n - 1$
13. return z

На линији један поставља се показивач z на минимални чвор. Уколико z не постоји, Фибоначијев хип је већ празан. Међутим, ако минимални чвор постоји, онда се пролази кроз свако дете чвора z и свако од њих се додаје кореној листи. Тиме они постају чворови корене листе, а њихови родитељи NIL. Када је завршен пролазак кроз сву децу, чвор z се избацује из корене листе (Фибоначијевог хипа H). Ако је z једнак свом десном потомку после линије шест, онда је чвор z једини чвор у кореној листи, и како он нема деце једино што преостаје јесте да се $H.min$ постави на NIL. Иначе, $H.min$ се поставља на десни потомак чвора z. То што је на њега постављен показивач $H.min$ не значи да је он обавезно и минимални чвор Фибоначијевог хипа, већ се он само тренутно налази ту.

Следећи корак је смањење броја стабала са хип својством у Ф-хипу, спајањем више чворова из корене листе, заједно са његовим потомцима, у више мањих хипова. Сједињавање се завршава оног тренутка када сваки чвор има степен који се разликује од степена било ког другог чвора из корене листе.

Процес сједињавања се врши наследећи начин:

1. Без смањења оптшоси може се претпоставити да за два чвора x и y из корене листе, истог степена важи:

$$x.klju\check{c} \leq y.klju\check{c}.$$

2. Уклонити y из корене листе, повезати чвор y са чвором x, тако да чвор y буде син чвора x. Тиме се степен чвора x повећава за један, а атрибут чвора $y.markiran$ постаје **false**.

За операцију **sjedinjavanja** користи се помоћни низ $A[0..D(H.n)]$ за чување информација о степенима чворова y корене листе. Ако је $A[i] = y$, то значи да је y корен са степеном i, односно $y.stepen = i$. Такође, овде се користи претпоставка да се зна горња граница максималног степена, односно $D(H.n)$.

Следећи псеудо код описује операцију сједињавања.

```
neka je A[0..D(H.n)] niz
for i = 0 to D(H.n)
    A[i] = NIL
za svaki čvor w u korenoj listi H
    x = w
    s = x.stepen
    while A[s]! = NIL
        y = A[s]
        if x.ključ > y.ključ
            zameni mesta čvorova x i y
        povezati y i x u Fibonačijevom hipu operacijom uvezati(H, x, y)
        A[s] = NIL
        s = s + 1
    A[s] = x
H.min = NIL
for i = 0 to D(H.n)
    if A[i]! = NIL
        if H.min == NIL
            kreiraj korenu listu Fibonačijevog hipa H koja sadrži samo A[i]
            H.min = A[i]
        else ubaciti A[i] u Fibonačijevu korenu listu
            if A[i].ključ < H.min.ključ
```

23. $H.min = A[i]$.

uvezati(H, y, x)

1. ukloniti y iz korene liste Fibonačijevog hipa H
2. postaviti da je y dete čvora x, uvećati stepen čvora x
3. atribut čvora y markiran se postavlja na **false**.

У прве три линије се алоцира и иницијализује низ, тако што сваки елемент добије вредност NIL. Низ A који се иницијализује заправо има двојаку улогу. Прва је да тренутно чува информације о степенима чворова док се не дође до стадијума када не постоје два чвора са истом вредношћу степена и друга да се на основу попуњеног низа реконструише структура. Код линија четири до четрнаест користи се for петља, како би се прошло кроз сваки чвор w корене листе. Индекс низа A представља степен чвора, тако да $A[s] = x$ значи да је степен чвора x s. Пролажењем кроз корену листу посматрамо текући чвор. Уколико у низу A не постоји чвор чија је вредност степена једнака текућем чвору, на индексу једнаком степену чвора, ставља се показивач на текући чвор, линија четрнаест. Међутим, ако чвор са степеном s, већ постоји у низу, онда важи $s = x.stepen = y.stepen$ па се због тога врши повезивање текућег чвора x и чвора y на следећи начин. Уколико важи да је вредност кључа чвора x већа од вредности кључа чвора y, онда се пре повезивања извши размена чворова, линија десет. Тиме x постаје y, а y x. Након тога, y се избацује из корене листе и постаје дете чвора x. Када се заврши тај процес повезивања, степен чвора x се увећава за један. Тиме се поступак у оквиру while петље наставља све док има чворова са истим степеном док се for петље наставља све док имамо чворова и кореној листи.

Већ поменута while петља, омогућава да се изврши сједињавање и осигурава да ће у низу постојати само један чвор са једним степеном.

Након завршетка for петље, остаје да се изврши линија петнаест која поставља $H.min$ на NIL, након које се врши реконструисање структуре на основу низа A.

Тиме је завршен опис операције **izbaci_min**. Ова операција умањује у линији 11 $H.n$ и у линији 12 враћа показивач на обрисани чвор z.

Анализа која следи показује да је амортизована цена извршавања операције **izbaci_min** Фибоначијевог хипа $O(D(n))$, где је n број чворова Фибоначијевог хипа.

Сложеност израчунавања се процењује на следећи начин. Пролазак кроз линије од 2 до 3 и од 16 до 23 изискује $D(n)$ операција. Остаје још да се анализирају линије од 4 до 14. Величина корене листе после позива операције сједини је највише $D(n) + s(H) - 1$, пошто у листи имамо $s(H)$ чланова (број стабала), минус један избачен чвор и плус број деце избаченог чвора (који постају коренови) који највише износи $D(n)$. Број итерација кроз while петљу зависи од величине корене листе, јер је сваки чвор из те листе повезан са другим и има их колико и елемената у листи. Према томе, укупна количина посла износи $D(n) + s(H)$, односно укупан посао је сразмеран $O(D(n)+s(H))$. С друге стране, потребно је израчунати промену потенцијала. Потенцијал пре операције је: $s(H) + 2m(H)$, а потенцијал након ње износи највише, $(D(n)+1)+2m(H)$, где је $D(n)+1$ горња граница дужине корене листе, а у току извођења операције не маркира се ниједан чвор. Дакле, амортизована цена извршавања операције је:

$$O(D(n) + s(H)) + ((D(n) + 1) + 2m(H)) - (s(H) + 2m(H))$$

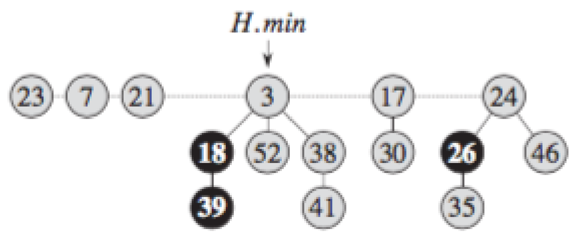
$$= O(D(n)) + O(s(H)) - s(H) = O(D(n))$$

Из чињенице да је

$$D(n) = O(\log n)$$

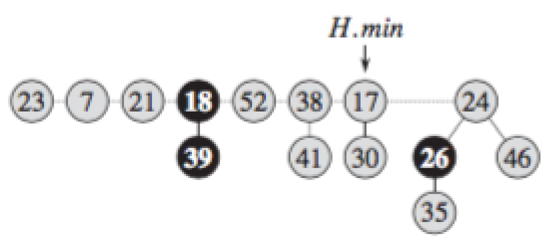
(следи из последице 6.0.4.1) следи да је амортизована цена извршавања операције $O(\log n)$.

У наредном делу следи илустрација операције **izbaci_min**.



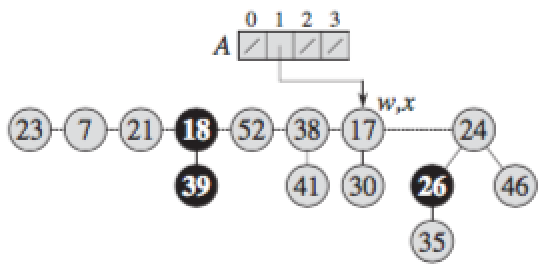
Слика 4.3: Пример Фибоначијевог хипа пре операције вађења минимума.

Чвор 3 се вади из хипа. Његова деца се додају кореној листи. Изглед овог дела је приказан наредном сликом.



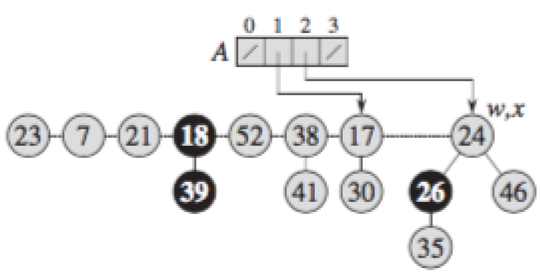
Слика 4.4: Пример Фибоначијевог хипа након првог дела операције

Вађењем минималног чвора, операција се не завршава. Неопходно је креирати низ A.

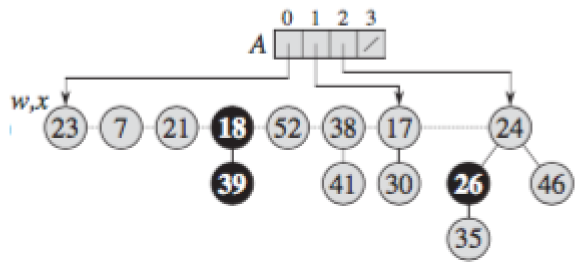


Слика 4.5: Креира се низ A чији елементи показују на чворове тако да је индекс чвора једнак степену чвора из корене листе

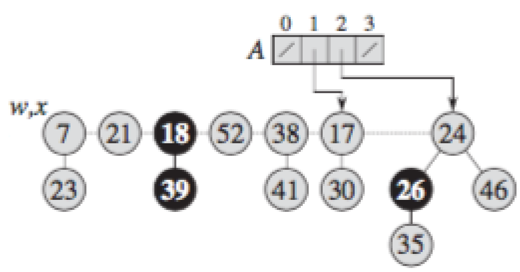
Вредности елемената низа су показивачи на чворове, а индекси су степени чворова из корене листе. Додавање у низ се започиње од $H.min$.



Слика 4.6: Иде се редом. Елемент са индексом два показује на чвор корене листе чији је степен два.

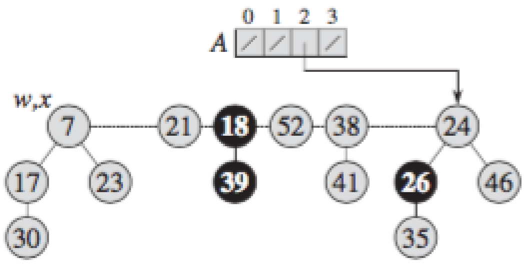


Слика 4.7: Елемент низа са индексом нула сада показује на чвор са степеном нула.

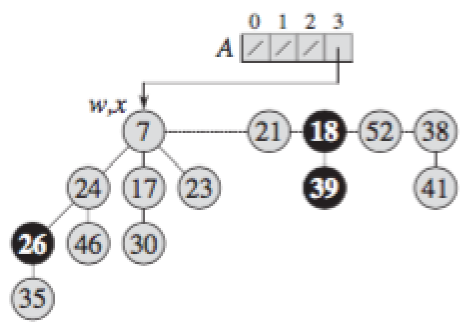


Слика 4.8: Следећи чвор је 7, са степеном 0, тако да се врши спајање чворова 7 и 23. Чиме се одмах и ажурира вредност степена чвора 7.

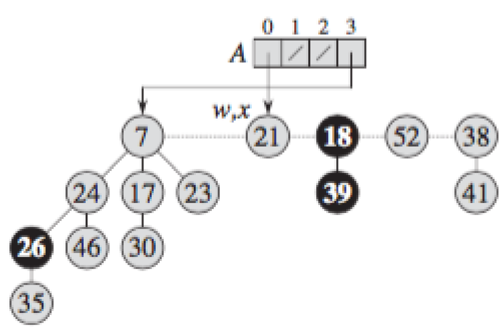
Када чвор који има исти степен као и неки други чвор из корене листе, онда се врши спајање, али тако да је испуњен *услов хипа.* У низу чвор 23, и чвор 7 имају исти степен. Тада чвор са мањим кључем постаје отац чвору са већом вредношћу кључа. У овом случају 7 постаје отац чвору 23. Овакав процес се наставља све док у кореној листи имамо чворове чији су степени исти.



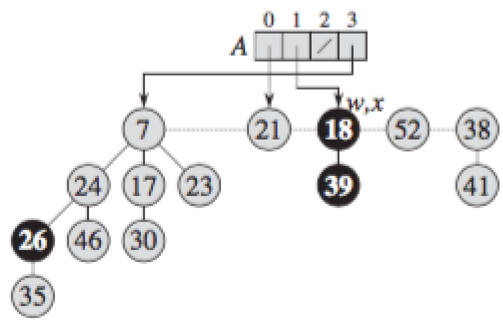
Слика 4.9: Овде се спајају чворови 7, са степеном 1 и чвор 17, исто са степеном 1.



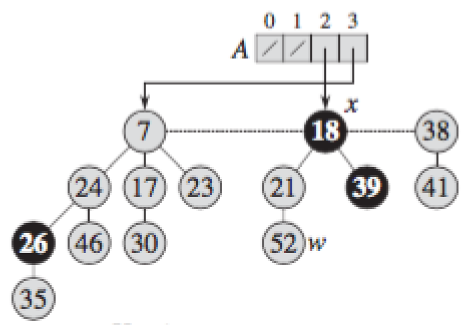
Слика 4.10: Опет долази до других сједињавања.



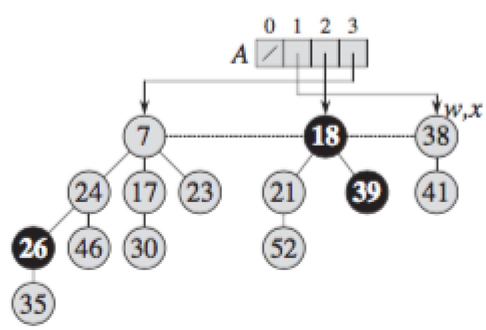
Слика 4.11: Степен чвора се увећава приликом сједињавања.



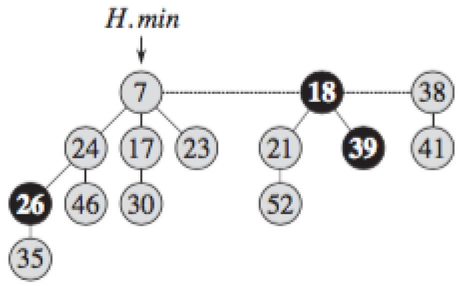
Слика 4.12: Стално се ажурирају степени.



Слика 4.13: Степени чвора 18 и 7 се разликују тако да не долази до њиховог спајања.



Слика 4.14: Сви степени се разликују.



Слика 4.15: На крају се ажурира показувач хипа на најмању врендост кључа.

Резултујући хип операције **izbaci_min**. Чворови имају различите степене.

4.7 Умањење кључа

Операцијом **umanji_kljuc**(H, x, k) се врши постављање новог кључа k у чвор x, притом се води рачуна да нови кључ не буде већи од кључа који се већ налази у чвору. Претпоставка са којом се креће у извршавање операције је да се брисањем чвора из повезане листе не мењају атрибути обрисаног чвора.

if $k > x.klju\check{c}$
  greška "novi ključ je veći od tekućeg ključa"
$x.klju\check{c} = k$
$y = x.otac$
if $y! = NIL$ and $x.klju\check{c} < y.klju\check{c}$
  odseci(H, x, y)
  kaskadno_odseci(H, y)
if $x.klju\check{c} < H.min.klju\check{c}$
  $H.min = x$

Операција **odseci** (H, x, y)

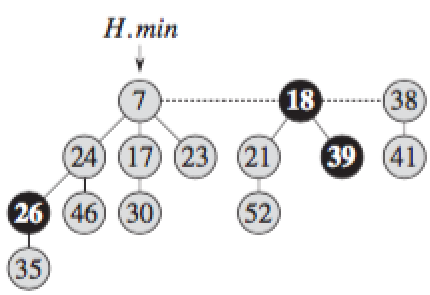
izbaciti čvor x koji je dete čvora y, smanjiti stepen čvora y
dodati x u korenu listu
$x.otac = NIL$

4. $x.markiran =$ **false**

Операција **kaskadno_odseci**(H, y)

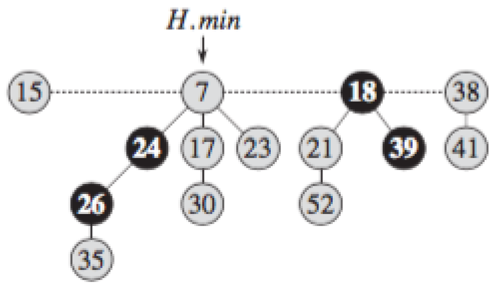
1. $z = y.otac$
2. **if** $z! = NIL$
3. **if** $y.markiran ==$ **false**
4. $y.markiran =$ **true**
5. **else**
6. odseci(H, y, z)
7. kaskadno_odseci(H, z)

Линије од један до три обезбеђују да нови кључ не буде већи од кључа чвора x, а онда га додељују чвору x. Уколико је x корен или $x.klju\check{c} \geq y.klju\check{c}$, где је y отац чвора x, онда нема структурних промене јер није нарушен минимални поредак хипа. Ово је тривијалан случај. Даље, линије четири и пет тестирају овај услов. Притом ако је нарушен поредак, алгоритам врши одсецање чвора x на линији шест. Операција **odseci** одсеца везу између x и y, правећи од чвора x корен. Како чвор x постаје корен онда он аутоматски постаје немаркиран. Након одсецања иде се у извршавање каскадног одсецања. Ту се посматра отац одсеченог чвора, y и отац његовог оца, $y.otac$. Ако је y немаркиран, онда се извршава линија четири и он постаје маркиран и излази се из функције. Ако је y маркиран, онда се он одсеца и рекурзивно се позива каскадно одсецање. Каскадно одсецање се позива све док постоје очеви који су маркирани, односно рекурзија се у оквиру каскадног одсецања завршава онда када се наиђе на корен или када нема више маркираних чворова. $H.min$ на крају ће или и даље показивати на оригиналну вредност кључа хипа или на чвор x са новом вредношћу кључа.



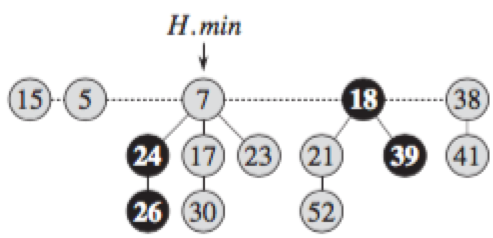
Слика 4.16: Слика Фибоначијевог хипа пре операције умањења кључа

На хип се примењује операција **umanjenje_kljuca** чвора 46 на нову вредност 15.



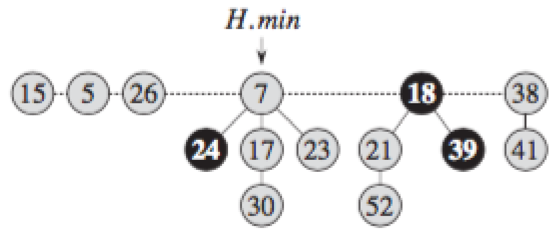
Слика 4.17: Смањује се кључ чвора и он се пребацује у коренску листу.

Извршава се промена вредности кључа, а потом се чвор 15 убацује у корену листу јер је мањи од вредности свога оца. Након тога, вршимо умањење чвора 35 на нову вредност 5.

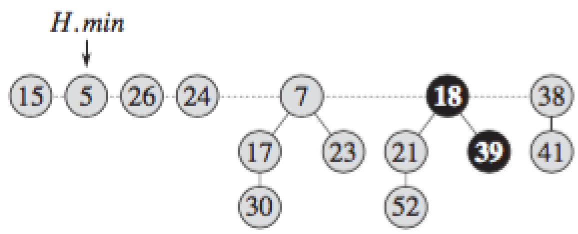


Слика 4.18: Следеће је смањење чвора са кључем 35 на 5.

Чвор 5 се помера у корену листу. Такође, и чвор са кључем 26 прелази у корену листу (јер је био маркиран) као и чвор са кључем 24 (исто је маркиран).



Слика 4.19: Чвор са промењеном вредоношћу (вредношћу 5) одлази у коренску листу. И долази до премештања осталих чворова који су маркирани.



Слика 4.20: Слика Фибоначијевог хипа након операције умањења кључа

Амортизована цена извршавања операције **umanji_ključ** је $O(1)$. Почећемо са анализом стварне цене извршавања. Операцији **umanji_ključ** је потребно $O(1)$, плус време за извшавање каскадног одсецања.

Претпоставимо да позивање операције **umanji_ključ** резултира позивањем **c** пута операције каскадног одсецања. Сваки позив каскадног одсецања се извршава за време $O(1)$. Према томе, стварна цена извршавања операције **umanji_ključ**, укључујући све рекурзивне позиве се извршава за $O(c)$.

У наставку следи промена потенцијала. Позивањем операције **odseci** на линији шест, операцијом **umanji_ključ**, креира се ново стабло чији је корен чвор x и атрибут чвора *markiran* се поставља на **false**. Сваки позив каскадног одсецања, осим последњег, одсеца маркирани чвор и поставља атрибут на **false**. Одсецање маркираног чвора доприноси томе да се одржи висина Ф-хипа. Након тога, Фибоначијев хип садржи $s(H) + c$ стабала (оригиналних $s(H)$ стабала, $c - 1$ стабло настало каскадним одсецањем и корен x), и највише $m(H) - c + 2$ маркираних, означених чворова ($c - 1$ су немаркирани применом каскадног одсецања и последњи позив каскадног одсецања може имати маркирани чвор). Промена у потенцијалу је највише, $((s(H) + c) + 2(m(H) - c + 2)) - (s(H) + 2m(H)) = 4 - c$.

Према томе, амортизована цена извршавања операције **umanji_ključ** је највише: $O(c) + 4 - c = O(1)$.

Из овога се извлачи закључак због чега се у дефиницију потенцијала два пута укључује број маркираних чворова. Када се маркирани чвор x одсеца каскадним одсецањем, он постане немаркиран, чиме се потенцијал смањује за два.

4.8 Брисање чвора

Следећи псеудокод брише чвор из Фибоначијевог хипа за амортизовано време $O(D(n))$, где је n број чворова хипа. Претпостављамо да не постоји кључ чија је вредност - ∞ у Фибоначијевом хипу.

obriši_čvor(H, x)

1. **umanji_ključ** $(H, x, -\infty)$
2. **izbaci_min** (H)

Операцијом **obriši** чвор x постаје минималан чвор тако што му се додели кључ са вредношћу - ∞. Након тога, операцијом **izbaci_min** се чвор x избацује из хипа. Амортизовано време извршавања ове операције је збир амортизованих времена за умањење кључа $O(1)$ и избацивање минималног чвора, односно $O(D(n))$. Како се показује да је $D(n) = O(\log n)$, онда је амортизовано време $O(\log n)$.

Глава 5

Фибоначијев хип у теорији и пракси

Са теоријског становишта, Фибоначијев хип је јако погодан за апликације које имају мали број позива операција **izbaci_min** и **obrisi** у односу на остале операције. На пример, неки алгоритми који се тичу проблема графова могу позивати **umanji_ključ** само једном по ивици графа. Код довољно густих графова операција **umanji_ključ** има сложеност извршавања $\Theta(1)$ (амортизовано време) за сваки позив што је велика предност у односу на исту операцију бинарног хипа која се у најгорем случају извршава за $\Theta(\log n)$. Сложеност операција Фибоначијевог хипа у односу на неке друге структуре се најбоље види приликом прављења минималног повезујућег стабла и приликом налажење најкраћег пута од чвора до свих осталих чворова.

Са практичне стране, константан фактор и програмска комлексност структуре Фибоначијевог хипа, чине је мање пожељном структуром од бинарног хипа за већину апликација, осим оних које раде са великом количином података. Због тога је Фибоначијев хип ипак значајнији у теорији него у пракси. Када би Ф-хип био једноставнији, бар што се структуре тиче, а да се притом задрже времена сложености операција, практична страна не би била доведена у питање.

Бинарни хип, као и Фибоначијев су неефикасни сто се тиче операције **pronadji**. Тражење елемента са датим кључем може потрајати. Због тога операције као што су **umanji_ključ** и **obrisi** као улазни параметар захтевају показивач на тражени елемент. Фибоначијев хип се може користити и као структура са свим својим операцијама, али и као структура у којој се могу чувати објекти апликације. Касније ће оваква употреба бити објашњена у делу примене, на примеру Хафмановог кода.

Глава 6

Процена горње границе за максимални степен

Да би се доказало да је амортизовано време за извршавање операција **izbaci_min** и **obriši** $O(\log n)$, потребно је да се покаже да је горња граница $D(n)$ степена било ког чвора у Фибоначијевом хипу $O(\log n)$. Испоставља се да је $D(n) \leq \log_\Phi n$, где је Φ златни пресек, дефинисан једнакошћу, $\Phi = (1 + \sqrt(5))/2 = 1.61803...$

За сваки чвор x Фибоначијевог хипа се дефинише **veličina**(x) која представља број чворова којима је x отац, и у тај број улази и сам чвор x.

Наредном лемом се показује да је величина чвора x експоненцијална функција степена чвора x.

Лема 6.0.1 *Нека је x било који чвор Фибоначијевог хипа, претпоставимо да је степен чвора x једнак k. Нека су y_1, y_2, ..., y_k деца чвора x у поретку у ком су повезана са x. Тада је $y_1.stepen \geq 0$ и $y_i.stepen \geq i-2$ за свако $i = 2, 3, ..., k$.*

Доказ. Тривијално y_1.stepen ≥ 0. За $i \geq 2$, када је y_i повезан са x, сви y_1, y_2, ..., y_{i-1} чворови су деца чвора x, зато мора да буде $x.stepen \geq i-1$. Зато што је сваки y_i повезан са x само ако је $x.stepen = y_i.stepen$, тада важи и y_i.stepen $\geq i-1$. Од тада, чвор y_i је изгубио највише једно дете, пошто би био одсечен од x, да је узгубио два детета. С тога може да се закључи да је y_i.stepen $\geq i-2$.

У наставку следи објашњење назива Фибоначијев хип. Фибоначијеви бројеви су дефинисани рекурентном релацијом и то на следећи начин:

$$F_k = \begin{cases} 0 & \text{ako je } k = 0 \\ 1 & \text{ako je } k = 1 \\ F_{k-1} + F_{k-2} & \text{ako je } k \geq 2. \end{cases}$$

Следећа лема даје другачији начин представљања F_k.

Лема 6.0.2 *За сваки цео број $k \geq 0$ важи:*

$$F_{k+2} = 1 + \sum_{i=0}^{k} F_i$$

Доказ. Доказ је заснован на математичкој индукцији по k. Када је $k = 0$, имамо :

$$1 + \sum_{i=0}^{0} F_i = 1 + F_0 = 1 + 0 = F_2$$

Сада претпоставимо да формула важи за

$$F_{k+1} = 1 + \sum_{i=0}^{k-1} F_i$$

и онда имамо:

$$F_{k+2} = F_k + F_{k+1} = F_k + \left(1 + \sum_{i=0}^{k-1} F_i\right) = 1 + \sum_{i=0}^{k} F_i$$

Лема 6.0.3 *За све целе бројеве $k \geq 0$, $(k+2)$ Фибоначијев хип задовољава следећу неједнакост:*

$$F_{k+2} \geq \Phi^k$$

Доказ. Доказ се ради на основу индукције по k. База индукције су случајеви за $k = 0$ и $k = 1$. Када је $k = 0$ имамо

$$F_2 = 1 = \Phi^0$$

и када је $k = 1$,

$$F_3 = 2 > 1.619 > \Phi^1$$

Индуктивна хипотеза. За $k \geq 2$ претпоставимо да важи

$$F_{i+2} > \Phi^i$$

где је: $i = 0, 1, ..., k-1$. Такође, Φ је позитиван корен једначине

$$x^2 = x + 1$$

Те имамо,

$$F_{k+2} = F_{k+1} + F_k \geq \Phi^{k-1} + \Phi^{k-2}$$

(на основу индуктивне хипотезе)

$$= \Phi^{k-2}(\Phi + 1) = \Phi^{k-2}\Phi^2$$

(на основу једнакости)

$$= \Phi^k$$

Лема 6.0.4 *Нека је x било који чвор у Фибоначијевом хипу чији је степен једнак k. Онда је* ***veličina(x)*** $\geq F_{k+2} \geq \Phi^k$*, где је*

$$\Phi = (1 + \sqrt{(5)})/2$$

.

Доказ. Нека је s_k минимална могућа величина било ког чвора степена k у Фибоначијевом хипу.

Тривијално, $s_0 = 1$ и $s_1 = 2$. Број s_k је највише **veličina**(x) и због додавања сина у чвор, не може се смањити величина чвора, тако да вредност s_k монотоно расте по k. Нека је чвор z, било који чвор Фибоначијевог хипа такав да је $z.stepen = k$, и veličina$(z) = s_k$. Пошто је $s_k \leq$ **veličina**(x), онда се израчунава доња граница **veličina**(x) тиме што се одреди доња граница s_k. Као што смо имали у леми, нека су y_1, y_2, ... , y_k синови чвора z, у поретку у ком су повезани са z. Да би се ограничио s_k, посматра се z и први син y_1, дато са:

$$\textbf{veličina}(x) \geq s_k \geq 2 + \sum_{i=2}^{k} s_{y_i.stepen}$$

$$\geq 2 + \sum_{i=2}^{k} s_{i-2}$$

Ова последња линија следи из леме 6.0.1 (где важи да је y_i.stepen $\geq i-2$) и због монотоности s_k важи да је $s_{y_i.stepen} \geq s_{i-2}$.

Након тога, доказујемо индукцијом по k да је $s_k \geq F_{k+2}$ за све ненегативне целе бројеве k. База индукције је: $k = 0$, $k = 1$ и то је тривијално. За индуктивну хипотезу претпоставимо да је: $k \geq 2$ и да је: $s_i \geq F_{i+2}$ за $i = 0, 1, ..., k-1$. Сада имамо:

$$s_k \geq 2 + \sum_{i=2}^{k} s_{i-2} \geq 2 + \sum_{i=2}^{k} F_i = 1 + \sum_{i=0}^{k} F_i = F_{k+2} \geq \Phi^k$$

Ова два последња реда следе на основу претходне две леме.

Овим је показано да је: *veličina*$(x) \geq s_k \geq F_{k+2} \geq \Phi^k$.

Последица 6.0.4.1 *Максимални степен $D(n)$ било ког чвора у Фибоначијевом хипу је $O(\log n)$.*

Доказ. Нека је x било који чвор Фибоначијевог хипа и k његов степен. На основу Леме, имамо да је $n \geq$ veličina(x) $\geq \Phi^k$. Применом основних особина алгоритма добијамо да је: $k \geq \log_\Phi n$. Тиме се показало да је максимални степен неког чвора $D(n) = O(\log n)$.

Глава 7

Програмска реализација, резултати и примене

У следећем делу су представљене програмске реализације експеримената и резултата. Операције са оба хипа, реализоване су у складу са описом у поглављу 2. Код је прилог рада као и програми којима се реализују експерименти. Експерименти су рађени у програмском језику C.

Бинарни хип је реализован преко низа. За смештање елемената хипа су коришћене локације од 1 до n, где је n текући број елемената хипа.

Дефиниција структуре бинарног хипа:

```
typedef struct element {
    unsigned long podatak;
} Element;

typedef struct binarniHip {
    long velicina;
    Element * element;
} BinarniHip;
```

Фибоначијев хип је реализован помоћу структуре која се састоји из показивача на оца, сина, лево и десно, атрибута који означава да ли је чвор маркиран, степена чвора и кључа, односно садржаја чвора.

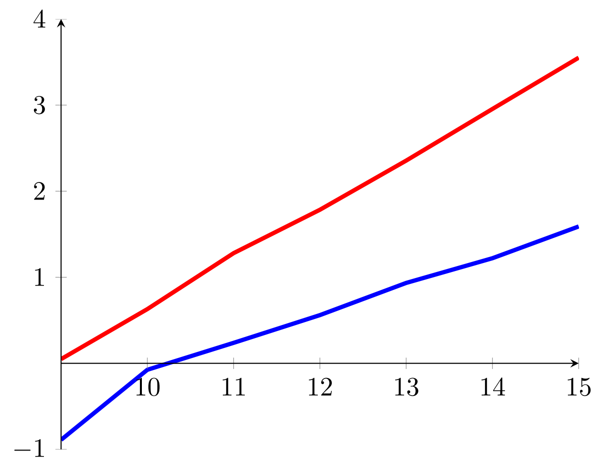
Дефиниција структуре Фибоначијевог хипа:

```
typedef struct fibonacijevHip {
    struct fibonacijevHip* sin;
    struct fibonacijevHip* otac;
    struct fibonacijevHip* levo;
    struct fibonacijevHip* desno;
    char markiran;
    unsigned long stepen;
```

```
    unsigned long kljuc;
} FibonacijevHip;
```

7.1 Упоређивање резултата за операције вађење минимума и упис

Фиксира се $N = 10^6$. Користи се генератор случајних бројева (31 - битних неозначених). Хип (бинарни и Фибоначијев) се пуни са 2^k случајних бројева, а онда се ради N случајно изабраних операција (независно изабраних са једнаком вероватноћом $1/2$) из скупа операција: вађење минимума и упис случајног броја. Мери се време. Резултат је низ времена за оба хипа, за $k = 9, 10, ..., 15$.



Слика 7.1: Упоређивање трајања времена извршавања операција вађење минимума и упис случајног броја код бинарног и Фибоначијевог хипа

Црвеном бојом је означен график који представља реализацију експеримента преко бинарног хипа, док је експеримент реализован Фибоначијевим хипом означен плавом бојом. X-координата представља фиксиране тачке, $k = 9, 10, 11, 12, 13, 14, 15$ док y-координата представља $logt$ (где је t у секундама), логаритам времена.

k	Бинарни хип (време извршавања у секундама)	Фибоначијев хип (време извршавања у секундама)
9	1.120	0.411
10	4.267	0.842
11	19.064	1.731
12	60.813	3.632
13	226.565	8.590
14	901.621	16.649
15	3556.827	38.902

Табела 7.1: Упоређивање времена извршавања експеримента на бинарном хипу и Фибоначијевом хипу

7.2 Упоређивање резултата за префиксни код

Креиран је програм који прави Хафманов (оптимални префиксни) код за азбуку од n симбола за коју су задати бројеви појављивања симбола, скуп $A = a_1, a_2, ..., a_n$. У експерименту се користе операције упис и вађење минимума. Нека је n дужина низа A и нека су учестаности реализације случајне променљиве са случајно равномерном расподелом из интервала $[0, 2^{20} - 1]$. Нека n узима вредности 2^k , где је: $k = 15, 16, ..., 28$. Нацртани график мери зависност $\log t$ (где је t време извршавања у секундама) времена од $\log n = k$. На тај начин су упоређене три операције (упис, смањење кључа и вађење минимума).

Хафманово кодирање је кодирање које се користи за компресију података који проналази оптимални систем кодирања објеката заснованом на релативној фреквенцији појављивања сваког објекта у колекцији. Код Хафмановог алгоритма креирају се симболи променљиве дужине који замењују улазне симболе. Што је вероватноћа појављивања симбола већа то је број бита којим се репрезентује симбол мањи. Кодирање, декодирање се врши тако што се пролази кроз стабло почевши од корена. У корену стабла се налази симбол са највећом вероватноћом појављивања, док се са мањом налазе даље, према листовима.

У наставку је дат псеудокод за Хафманов алгоритам. Нека је S ниска знакова која се појављује у тексту, F фреквенције и T кодно стабло.

1. ubaciti sve znake u hip prema njihovim frekvencijama
2. while H neprazan
3.   **if** H sadrži samo jedan znak X
4.   formiraj stablo T koje ima samo koren, čvor X
5.   **else**

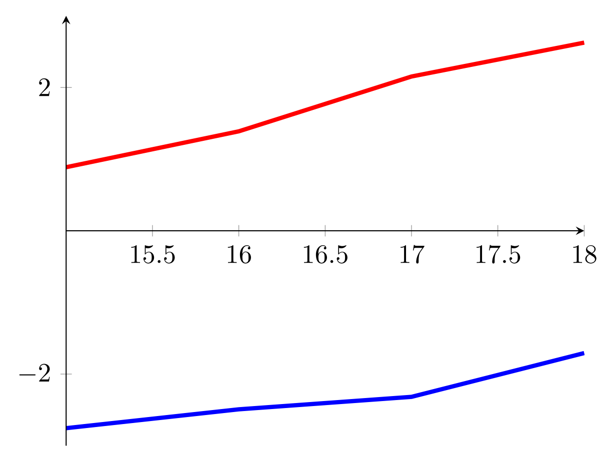
6. skini sa hipa H znakove X i Y sa najmanjim frekvencijama F_x, F_y

7. zameni X i Y novim znakom Z, sa frekvencijom $F_z = F_x + F_y$

8. ubaci Z u hip H

9. poveži X i Y tako da budu sinovi u T_i; Z još nema oca

Структура хипа је проширена са два нова атрибута, а то су: фреквенција и реч. У атрибуту фреквенција се чува информација о броју појављивања симбола, док се симбол налази у атрибуту реч. Алгоритам почиње тако што се ваде два елемената са најмањим фреквенција, рачуна се сума њиховог броја појављивања, конкатенирају се симболи и онда се те две нове информације, збир фреквенција и спојени симболи, убацују у структуру. Поступак се наставља све док хип не остане са једним чланом.



Слика 7.2: Упоређивање трајања времена проналажења префиксног кода код бинарног и Фибоначијевог хипа

Црвеном бојом је означен график који представља реализацију експеримента преко бинарног хипа, док је експеримент реализован Фибоначијевим хипом означен плавом бојом. X-координата представља фиксиране тачке, $k = 9, 10, 11, 12, 13, 14, 15$ док y-координата представља $logt$ (где је t у секундама), логаритам времена.

k	Бинарни хип (време извршавања у секундама)	Фибоначијев хип (време извршавања у секундама)
15	7.718739	0.001751
16	24.452	0.003218
17	142.25	0.004812
18	422.88	0.019608

Табела 7.2: Упоређивање времена извршавања експеримента на бинарном хипу и Фибоначијевом хипу

7.3 Упоређивање резултата за минимално повезујуће стабло

Нека су $N = 10$, $k = 5, 6, ...N$, и нека је $n = 2^k$, број чворова. Нека је m број грана, $m = \alpha n(n-1)/2$. Нека је случајна симетрична матрица n x n, са $2m$ јединица, без јединица на дијагонали. Матрица се пуни тако што се праве случајни парови (i, j), $1 \leq i < j \leq n$ док их се не скупи m различитих. Тиме се добија матрица повезаности . Ако се добије пар који је већ направљен онда се он одбацује. Након креирања матрице повезаности, на граф се примењује алгоритам за добијање минималног повезујућег стабла, користећи један, односно други хип. Тиме се врши поређење хипова у зависности од броја k и α.

Под минимално повезујућим стаблом се подразумева стабло које садржи све чворове једног графа тако да сума цена грана буде минимална.

Псеудокод за проналажење минималног повезујућег стабла је приказан у наставку. Нека је T празан скуп. T је минимално повезујуће стабло. G тежински неусмерен граф. Сваки чвор графа може бити означен или не, у зависности од тога да ли је посећен и сваки чвор има своју цену.

1. na početku T je prazan skup
2. za sve čvorove w
3. $\quad w.oznaka =$ **false**
4. $\quad w.cena = \infty$
5. нека је (x, y) грана са најмањом ценом у G
6. $x.oznaka =$ **true**
7. za sve grane (x, z)
8. $\quad z.grana = (x, z)$ (grana najmanje cene)
9. $\quad z.cena = cena(x, z)$ (cena grane $z.grana$)

while postoji neoznacen cvor

neka je w neoznaceni čvor sa najmanjom vrednošću $w.cena$

if $w.cena = \infty$

stampaj "G nije povezan"

else

$w.oznaka =$ **true**

dodaj $w.grana$ u T ovde se sada popravljaju cene neoznačenih čvorova povezanih sa w

for sve grane (w, z)

if not $z.oznaka$

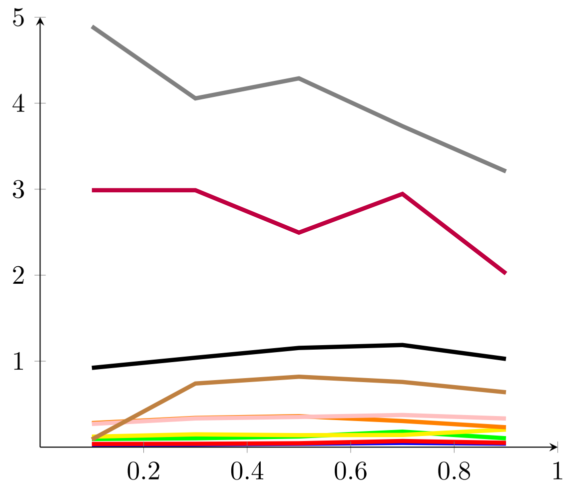
if $cena(w, z) < z.cena$

$z.grana = (w, z)$

$z.cena = cena(w, z)$

Прва изабрана грана је грана са најмањом ценом. Приликом проласка кроз гране, гледа се да је грана повезана са T и са неким чвором ван T. T је на почетку празан скуп. За сваки чвор који се налази ван T се памти минимална цена. У свакој итерацији се пролази кроз гране, и убације чвор са минималном ценом. Потом се послатрају гране из тог чвора и ако постоји чвор са ценом мањом од цене већ убаченог чвора,

У свакој итерацији се бира грана најмање цене и повезује се са одговарајућим чвором w са T. Затим се проверавају све гране суседне чвору w. Ако је цена неке такве гране (w, z) где z не припада T мања од цене тренутно најјефтиније познате гране до z онда се преправља цена чвора z.



Слика 7.3: Упоређивање трајања времена проналажења минималног повезујућег стабла код бинарног и Фибоначијевог хипа

Црвени, жути, розе, црни и сиви графици представљају редом резултате експеримента реализованог преко бинарног хипа. Док су плавом, зеленом, наранџастом, браон и тамно љубичастом бојом представљени графици који репрезентују податке добије коришћењем Фибоначијевог хипа. На x-оси је представљен параметар алфа који узима следеће вредности $\alpha = 0.1$, $\alpha = 0.3$, $\alpha = 0.7$, $\alpha = 0.9$. Парови црвени и плави, зелени и жути, наранџасти и розе, браон и црни, тамно љубичасти и сиви су приказани за следеће фиксиране вредности: $k = 5$, $k = 6$, $k = 7$, $k = 8$ и $k = 9$.

Глава 8

Закључак

Овај рад је посвећен хип структури података. На почетку је представљена структура бинарни хип. Описане су програмска имплементација и могуће операције са овом структуром података. Потом је у раду акценат стављен на Фибоначијев хип. Поред објашњења структуре, детаљно су објашњене операције креирања Фибоначијевог хипа, убацивања чвора, проналажења минималног кључа и спајања два различита Фибоначијева хипа. Објашњене су и операције вађења минималног чвора, умањења кључа и брисања одређеног чвора. На самом крају су у серији експеримената упоређени резултати при манипулацији великим скупом података коришћењем структуре бинарног хипа и структуре Фибоначијевог хипа.

Комплексност имплементације Фибоначијевог хипа га чине не превише коришћеном структуром у пракси. Међутим, у ситуацијама када се манипулише великом количином података и алгоритми за манипулацију не користе често операције **izbaci_min** и **obrisi**, коришћење Фибоначијевог хипа омогућава много боље перформансе алгоритама него коришћење бинарног хипа. Остаје нада да ће се, у будућности, са повећањем количине података која треба да се анализира, јавити више прилика у којима ће коришћење Фибоначијевог хипа донети значајни бољитак перформанси.

Библиографија

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein Introduction to Algorithms. *MIT Press, Cambridge Massachusetts*, 2009.

[2] Miodrag Živkovic Algoritmi. *Matematički fakultet, Beograd*

[3] Michael L. Fredman, Robert E. Tarjan Fibonacci Heaps and Their Uses in Improved Network Optimization Algorithms. June, 1986 https://www.cs.princeton.edu/courses/archive/fall03/cs528/handouts/fibonacci%20heaps.pdf

[4] Fibonacci Heap in C, Code Review, Stack Overflow. http://codereview.stackexchange.com/questions/117965/fibonacci-heap-in-c