Strukture Podataka i Algoritmi

Lekcija 7

leto 2019/2020

Prof. dr Branimir M. Trenkić

Skupovi kao ATP

- ***Skup*** (engl. ***set***) – kao matematički model se koristi ***u mnogim algoritmima***
- ***Skup*** predstavlja kolekciju elemenata (ili članova):
  - Svi elementi skupa su međusobno ***različiti***
  - Unutar kolekcije se ne zadaje ***nikakvo*** eksplicitno linearno ili hijerarhijsko ***uređenje***
  - Svaki ***element skupa*** može biti ili ***drugi skup*** ili ***primitivni element*** koji se naziva ***atom***
  - Atomi su najčešće ***primitivni tipovi*** podataka (celi brojevi, karakteri ili stringovi) i ***svi elementi u skupu su istog tipa***

Skupovi kao ATP

- U praksi se uvodi jedna ***pretpostavka***:
- Postoji (***implicitna, prirodna***) ***relacija totalnog uređenja*** među atomima skupa
- Ova relacija se označava sa „**<**“ a čita kao ***„manje od“***
  - Ova relacija omogućuje uvođenje neke vrste ***linearnog uređenja***, u smislu da se može tumačiti i kao „prethodi“

Skupovi kao ATP

- ***Linearno uređenje*** koje se postiže ***relacijom <*** na skupu ***S*** zadovoljava ***dva pravila***:

1. Za bilo koja dva elementa ***a*** i ***b*** skupa ***S***, važi tačno jedna od sledećih relacija

   a < b ili ***a > b***

2. Za svako ***a***, ***b*** i ***c*** iz skupa ***S*** važi,

   ako je ***a < b*** i ***b < c*** tada je ***a < c***

   (***zakon tranzitivnosti***)

Skupovi kao ATP

- ***Skup*** se definiše kao apstraktni tip podataka ***sa operacijama*** koje su ***uobičajne u matematici***
- Sve operacije možemo svrstati u ***dve kategorije***:

1. Operacije koje definišu ***odnos među skupovima*** (***pripadnost***, ***podskup***, ***nadskup***, ...),
2. Elementarne ***matematičke operacije*** (***unija***, ***presek***, ***razlika***, ...)

Operacije nad skupovima

- *Kreiranje praznog skupa*. ***set-make-null(A)*** - Skup je ***prazan*** ako ne sadrži ni jedan element

- *Dodavanje novog elementa u skup*.

set-insert(x, A)

- ***dodaje*** element ***x*** u skup ***A***, tj. transformiše skup ***A*** u skup ***A U {x}***. Ako x već postoji u skupu A, tada operacija nema efekta nad skupom A

Operacije nad skupovima

- *Uklanjanje elementa iz skupa*

set-delete(x, A)

– ***izbacuje*** element x iz skupa A, tj. transformiše skup ***A*** u skup ***A \ {x}***. Ako x nije u skupu A, tada operacija nema efekta nad skupom A

Operacije nad skupovima

- *Operacija pripadnosti*, ***set-member(x, A)*** – operacija vraća true (tačno) ako $x \in A$, inače vraća false (netačno)
- *Određivanje najmanjeg ili najvećeg elementa skupa*, ***min(A)*** ili ***max(A)*** – operacija vraća najmanji, odnosno, najveći element skupa A u smislu < uređenja
- *Podskup skupa*, ***subset(A, B)*** – operacija vraća true (tačno) ako važi $A \subseteq B$, inače vraća false (netačno)

Operacije nad skupovima

- *Unija dva skupa*, ***set-union(A, B, C)*** – operacija kreira skup C za koji važi $C = A \cup B$

- *Presek dva skupa*, ***set-intersection(A, B, C)*** – operacija kreira skup za koji važi $C = A \cap B$

- *Razlika dva skupa*, ***set-difference(A, B, C)*** – operacija kreira skup C jednak razlici skupova A i B, tj. ***A \ B***

Implementacija Skupa

Implementacija skupa pomoću niza:

- Ukoliko su svi skupovi od interesa ***podskupovi*** nekog relativno malog ***„univerzalnog skupa“***
- Postoji vrlo efikasna implementacija skupa nizom, koji se naziva ***karakteristični niz*** ili ***bit-vektor***
- Karakteristični niz je niz čiji elementi imaju samo ***vrednosti 0*** ili ***1***
- ***Sledi formalniji prikaz*** ove implementacije

Implementacija Skupa

Implementacija skupa pomoću niza:

- Neka je dat ***univerzalni skup*** od n elemenata, tj. ***U = {x_1, x_2, ..., x_n}***, ***|U| = n***
- Ako sa ***S*** označimo ***skup od interesa***, tako da važi $S \subseteq U$, možemo ga predstaviti karakterističnim nizom dužine n, na sledeći način:
- $(\forall x_i \in U)$ ako je $x_i \in S$ tada ***i-ti element*** karakterističnog niza ***ima vrednost 1***;
- ako je $x_i \notin S$ tada ***i-ti*** element karakterističnog niza ***ima vrednost 0***

Implementacija Skupa

Implementacija skupa pomoću niza: ***Primer***

- Neka je dat ***univerzalni skup***
  U = {Crvena, Plava, Žuta, Crna, Bela}, ***|U| = 5***
- Vrednost pojedinih elemenata niza, odnosno poredak boja:

Poredak	Boja
1	Crvena
2	Plava
3	Žuta
4	Crna
5	Bela

Implementacija Skupa

Implementacija skupa pomoću niza: ***Primer***

- ***Dužina*** univerzalnog skupa ***je 5*** - čime je određena i ***dužina karakterističnog niza***
- ***Podskupove od interesa*** čine boje koje ne pripadaju univerzalnom skupu boja a čiji su elementi boje iz univerzalnog skupa - ***kojima je moguće napraviti datu boju***

Implementacija Skupa

Implementacija skupa pomoću niza: ***Primer***

- ***skup Zelena*** = ***{Plava, Žuta}*** – ***karakteristični niz G*** kojim se predstavlja skup Zelena je oblika:

Poredak	Boja
1	Crvena
2	Plava
3	Žuta
4	Crna
5	Bela

Rešenje:

G:

0	1	1	0	0

Implementacija Skupa

Implementacija skupa pomoću niza: ***Primer***

- ***skup Braon*** = ***{Crvena, Plava, Žuta, Crna}*** – ***karakteristični niz B*** kojim se predstavlja skup Braon je oblika:

B:

1	1	1	1	0

Poredak	Boja
1	Crvena
2	Plava
3	Žuta
4	Crna
5	Bela

- ***prazan skup*** – odgovarajući ***karakteristični niz E*** je sa elementima:

E:

0	0	0	0	0

Implementacija Skupa

Implementacija skupa pomoću niza: ***Algoritam***

- Algoritam kojim se realizuje skupovna ***operacija unije*** dva skupa

set-union(a,b,c)

Implementacija Skupa

Implementacija skupa pomoću niza: ***Algoritam***

- Algoritam kojim se realizuje skupovna ***operacija unije*** dva skupa:

```
// Ulaz: karakteristični nizovi a i b skupova A i B

// Izlaz: karakteristični niz c koji predstavlja skup C

algorithm set-union(a, b, c)

    for i = 1 to n do

        if ((a[i] == 1) || (b[i] == 1))  then

            c[i] = 1;

        else

            c[i] = 0;

    return c;
```

Implementacija Skupa

Implementacija skupa pomoću niza: ***Algoritam***

- Algoritam kojim se realizuje skupovna ***operacija preseka*** dva skupa:

  set-intersection(a,b,c)

- Algoritam kojim se realizuje skupovna ***operacija razlike*** dva skupa:

  set-difference(a,b,c)

Domaći zadatak!

Implementacija Skupa

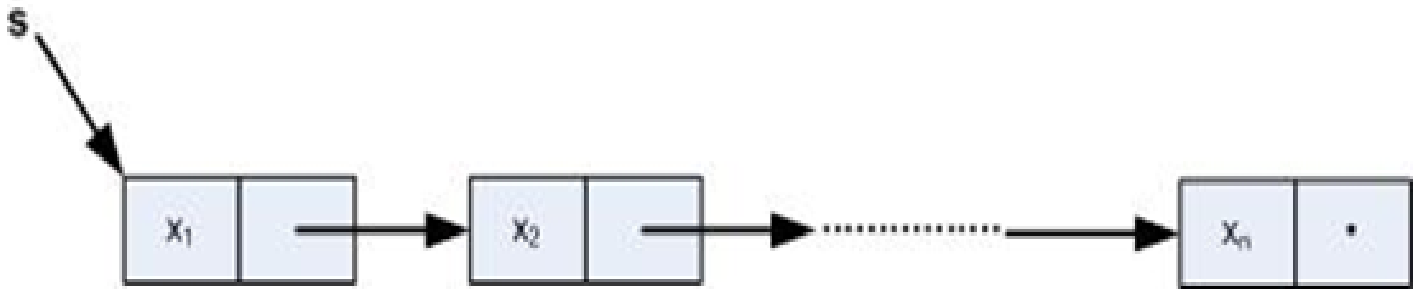
Implementacija skupa pomoću listi:

- ***Statička*** implementacija skupa ***se odlikuje jednostavnošću***
- Teško je realizovati ***univerzalno rešenje*** na ovaj način
- ***Svako*** naknadno ***proširenje*** (recimo, dužine univerzalnog skupa) prouzrokuje ***probleme u realizaciji***
- Da bi se prevazišli tipični problemi izazvani statičkom implementacijom primenjuje se realizacija bazirana ***na dinamičkim strukturama***

Implementacija Skupa

Implementacija skupa pomoću listi:

- Sledi implementacija skupa korišćenjem ***jednostruko povezane liste***
- Skup u opštem obliku $S = \{x_1, x_2, ..., x_n\}$ možemo predstaviti jednom ***jednostruko povezanom listom***,



Implementacija Skupa

Implementacija skupa pomoću listi: ***Algoritam***

- U slučaju elementarnih skupovnih operacija - situacija je ***malo komplikovanija***
- ***Algoritmi*** koji realizuju te operacije u ovoj implementaciji skupa, ***su složeniji*** i ***koriste više operacija nad listama*** u svojoj realizaciji

- Mi ćemo ovde predstaviti ***algoritam koji realizuje operaciju unije dva skupa***, tj. operaciju

$$C = A \cup B$$

Implementacija Skupa

Implementacija skupa pomoću listi: ***Algoritam***

```
// Ulaz: a i b - spoljašnji pokazivači na liste koje predstavljaju skupove A i B

// Izlaz: lista c koja predstavlja skup C koji je rezultat unije nad skupovima A i B

algorithm set-union(a, b, c)

    list-copy(a, c);

    for za svaki element x iz skupa B do

        if (list-search(a, x) == null)  then

            list-insert(x, null, c);

    return c;
```

list-head-insert(x, c)

Implementacija Skupa

Implementacija skupa pomoću listi: ***Algoritam***

- Objašnjenje:
  - ***Inicijalizacija rezultujućeg skupa C*** funkcijom ***list-copy*** - kopira sadržaj jednog skupa u drugi skup
  - Za ***sve elemente*** skupa B se ***provarava*** njihova ***pripadnost skupu A***
    - Ako tekući element ***ne pripada*** skupu A tada se taj element ***dodaje skupu C***
      - Ovo se realizuje operacijom dodavanja novog čvora na početak liste
    - U suprotnom, ne ostvaruje se ***nikakav efekat*** na skup C

Implementacija Skupa

- *Implementacija skupa pomoću listi*: ***Algoritam***
- Zapažanja:
- Ako se ***pretpostavi uređenost*** skupova, algoritam ***set-union remeti redosled*** rezultujuće liste (skupa)
- U listu ***c***, nakon kopiranja sadržaja liste ***a*** (koja je sortirana), dodaju se pojedini elementi liste ***b*** ali ***samo na početak liste*** ***c*** čime se ***redosled*** elemenata ***remeti***
- U tom slučaju, logički je u implementaciji ***primeniti sortirane liste***

Implementacija Skupa

- *Implementacija skupa pomoću listi*: ***Algoritam***
- Zapažanja:
- Izvršimo ***analizu vremena*** izvršavanja algoritma ***set-union***
- Predpostavimo, ***|A| = n*** i ***|B| = m***
- ***Kopiranje skupa A*** je vremenski proporcionalno broju elemenata skupa A, tj. jednako je ***O(n)***
- ***for petlja*** se izvršava ***m puta*** i u svakoj iteraciji se izvršava operacija ***pretrage liste dužine n*** – vreme izvršavanja kompletne petlje je proporcionalno proizvodu m·n, t.j. ***O(nm)***

Implementacija Skupa

- *Implementacija skupa pomoću listi*: ***Algoritam***
- Zapažanja:
- Izvršimo analizu vremena izvršavanja algoritma ***set-union***
- Ukupno vreme potrebno za izvršavanje algoritma je ***O(n) + O(mn) = O(mn)***
- Dakle, vreme izvršavanja algoritma ***set-union*** u ovoj implementaciji je ***proporcionalno proizvodu broja elemenata*** skupova A i B

Implementacija Skupa

- *Implementacija skupa pomoću listi*:
- Zapažanja:
- Kada bi lista bila uvek ***sortirana*** – operacije kao što su:
  - ***set-insert(x,S)***
  - ***set-delete(x,S)***
  - ***set-member(x,S)***
- Pretraživale bi listu ***samo do*** prvog elementa koji je ***veći od x***
- Prema tome, vreme izvršavanja ***u najgorem slučaju*** je jednako ***O(n)***

Implementacija Skupa

- *Implementacija skupa pomoću **sortiranih** listi*:
- U ovoj implementaciji ***najvažnije je očuvanje poredka*** elemenata u listi
- Da bi se to postiglo potrebno je ***modifikovati pojedine operacije nad listama***
  - Operacije ***dodavanja***, ***uklanjanja*** i ***pretrage***
- ***Kao primer*** algoritamske realizacije skupovnih operacija korišćenjem sortiranih listi u implementaciji skupova navešćemo algoritam za ***dodavanje novog elementa u dati skup***

Implementacija Skupa

- *Implementacija skupa pomoću **sortiranih** listi*:
- U ovom slučaju, mora se

1. ***Locirati pozicija u listi nakon*** koje treba ***ubaciti novi čvor*** tako da lista i dalje ***ostane sortirana***
2. Tek ***nakon toga*** vrši se ***dodavanje*** novog elementa u listu

Implementacija Skupa

- *Implementacija skupa pomoću **sortiranih** listi*:

```
// Ulaz: sortirana lista s koja predstavlja skup

// Izlaz: lista s sa čvorom x na propisanoj poziciji

algorithm set-insert(x, s)

    p = s;          // dodatni pokazivač na početak liste

    if ((p == null) || (p.ključ > x.ključ))  then

        list-insert(x, null, s);    // dodati čvor x na početku liste

    else

        while ((p != null) && (p.ključ < x.ključ))  do

            q = p;

            p = q.sled;

        if ((p == null) || (p.ključ != x.ključ))  then

            list-insert(x, q, s);   // dodati čvor x iza čvora q

    return s;
```

lista prazna ili element na glavi veći od onog koji se dodaje

list-head-insert(x, c)

štamparska greška u knjizi!

Implementacija Skupa

- *Implementacija skupa pomoću **sortiranih** listi*:
- Razmotrimo sada operaciju ***unije dva skupa*** u ovoj implementaciji
- ***Skupovi A i B*** su predstavljeni pomoću ***dve sortirane liste*** (spoljašnjim pokazivačima ***a*** i ***b***)
- Sam algoritam možemo opisati na sledeći način:
- U ***prvom delu*** algoritma, kako bi sačuvali podatke skupova A i B u izvornom obliku (redosledu) ***pravimo*** njihove ***kopije*** (liste) nad kojima algoritam faktički radi

Implementacija Skupa

- *Implementacija skupa pomoću **sortiranih** listi*:
- ***Ključni deo*** algoritma je sastoji od prostog ***upoređivanja početnih čvorova*** listi kopija
- Ako su ključevi različiti
  - ***Manji*** od njih ***uklanjamo*** iz odgovarajuće liste i ***dodajemo ga na kraj rezultujuće liste***
- Ako su ključevi jednaki
  - ***Oba uklanjamo*** iz odgovarajućih listi i ***dodajemo ga na kraj rezultujuće liste***

Implementacija Skupa

- *Implementacija skupa pomoću **sortiranih** listi*:
- ***Algoritam – I deo***

Dok su obe liste neprazne!

```
algorithm set-union(a, b, c)
    list-copy(a, l1); list-copy(b, l2); list-make(c);
    while ((l1 != null) && (l2 != null))  do
        if (l1.ključ == l2.ključ)  then
            x = list-delete(null, l1);
            x = list-delete(null, l2);
        else if (l1.ključ < l2.ključ)  then
            x = list-delete(null, l1);
        else
            x = list-delete(null, l2);
        list-insert(x, tail(c), c);
```

vraća pokazivač na poslednji čvor

Implementacija Skupa

- *Implementacija skupa pomoću **sortiranih** listi*:
- ***Algoritam – II deo***

Jedna od listi je prazna! Koja?

```
while (l1 != null)  do
        x = list-delete(null, l1
        list-insert(x, tail(c), c);
while (l2 != null)  do
        x = list-delete(null, l2);
        list-insert(x, tail(c), c);
return c;
```

Kraj algoritma

Implementacija Skupa

- *Implementacija skupa pomoću **sortiranih** listi*:
- Izvršimo ***analizu vremena*** izvršavanja algoritma ***set-union***
- Predpostavimo, ***|A| = n*** i ***|B| = m***
- ***set-union*** **-** Prolazi se kroz obe liste, tj. vreme je proporcionalno n + m, tj. ***O(n + m)***

Ukupno vreme = vreme sortiranja + vreme set-union

- ***Ukupno vreme*** = O(nlogn + mlogm) + O(n + m)

  = O(nlogn + mlogm)