Konstrukcija i analiza algoritama 2

NP kompletni problemi - priprema za ispitne rokove

Napomena: Zadaci koji slede rešeni su detaljnije nego što to ispit zahteva jer su predvidjeni za učenje i vežbu.

1. Klika je podgraf grafa takav da su u njoj svi čvorovi medjusobno povezani. Problem klika glasi: Da li u datom grafu postoji klika veličine k? Nezavisan skup je podskup čvorova grafa takav da nikoja dva čvora iz tog skupa nisu medjusobno povezana granom. Problem nezavisnog skupa glasi: Da li u datom grafu postoji nezavisan skup veličine m?

   (a) **(2 poena)** Kakav je odnos izmedju ova dva problema?

   (b) **(2 poena)** Dati pohlepan aproksimativni algoritam za problem pronalaženja maksimalne klike u grafu i njegov pseudokod.

   (c) **(2 poena)** Dati brute-force algoritam za problem pronalaženja najvećeg nezavisnog skupa i njegov pseudokod.

   (d) **(4 poena)** Pokrivač čvorova je podskup čvorova grafa takav da su sve grane tog grafa incidentne barem jednom čvoru iz tog podskupa. Problem pokrivača čvorova glasi: Da li za dati graf postoji pokrivač čvorova veličine l? Pokazati da je problem klika NP kompletan problem. Za svodjenje koristiti problem pokrivača čvorova.

   REŠENJE:

   (a) Da bismo redukovali problem pronalaska klike veličine k u datom grafu $G = (V, E)$ na problem pronalaženja nezavisnog skupa u istom grafu, potrebno je da posmatramo komplementarni graf datog $G' = (V', E')$ pri čemu je $V' = V$ a u E' se nalaze sve moguće grane izmedju čvorova u V koje nisu prisutne u E.

   - Ako postoji nezavisan skup veličine k u komplementarnom grafu G', to implicira da nikoja dva od tih k čvorova nisu medjusobno povezana granom u G'. Odatle, po načinu na koji je G' konstruisan implicira da postoji veza izmedju svaka dva od ovih čvorova u G odnosno oni grade kliku veličine k u grafu G.
   - Slično, ako postoji klika veličine k u grafu G, to implicira da su svaka dva od tih k čvorova povezana granom koja se nalazi u G. Te grane se ne nalaze u G' odnosno, u G' nikoja dva od tih

čvorova nisu povezana granom, te ovi čvorovi grade nezavisan skup veličine k u grafu G'.

(b) Pošto se traži pohlepan aproksimativni algoritam, ideja je da se u svakom momentu bira ono što se čini kao najbolji izbor za taj "lokalni" momenat, bez razmišljanja o konačnom ishodu. Pošto je nama cilj da pronadjemo maksimalnu kliku (a u klici su svi čvorovi medjusobno povezani), ideja koja se nameće jeste da posmatramo čvorove sa što većim stepenom, jer što su više povezani, veća je šansa da će biti deo velike klike. Nakon što sortiramo čvorove po stepenu opadajuće, idemo od čvora sa najvećim stepenom ka čvorovima sa nižim stepenom, i dodajemo jedan po jedan po sledećem kriterijumu: da bi čvor bio dodat u potencijalno rešenje, potrebno je da bude povezan sa svim prethodno dodatim čvorovima. Sledi pseudokod:

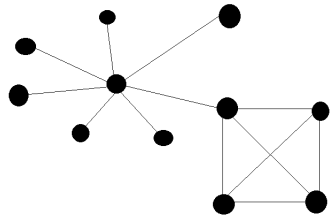
Algorithm 1 GreedyKClique

Require: Graph $G = (V, E)$, integer k
Ensure: *"YES" if a k-clique is found, otherwise "NO"*

```
ordered_vertices ← sort(V) by descending degree
C ← ∅                                              ▷ Candidate clique
for v ∈ ordered_vertices do
    if isAdjacentToAll(v, C) then
        C ← C ∪ {v}
    end if
    if |C| ≥ k then
        return "YES, clique of size k found"
    end if
end for
if |C| < k then
    return "NO, k-clique not found by greedy approach"
end if
```

Dati algoritam je aproksimacioni, tj. ne daje uvek tačno rešenje, što se može videti na sledećem primeru:

Na početku će u potencijalno rešenje biti dodat centar zvezde (jer on



ima najveći stepen). Nakon toga će se dodati čvor povezan sa centrom

zvezde koji ima najveći stepen - gornji levi ugao kvadrata. Dalje, po našem algoritmu, tražimo sledeći čvor sa najvećim stepenom povezan sa oba ova čvora. Ali takvog nema! Što će reći, dobijamo da je najveća klika u ovom grafu veličine 2, a to nije tačno rešenje (kvadrat sa dijagonalama je kompletan graf i u njemu je klika veličine 4).

(c) Najnaivniji mogući brute-force algoritam koji možemo da damo za ovo je sledeći: Naime, s obzirom da nam se traži maksimalan nezavisan skup čvorova u našem grafu, možemo da isformiramo sve podskupove čvorova našeg grafa i da proverimo za svaki da li su u njemu medjusobno nezavisni čvorovi. Pošto nam se traži najveći nezavisan skup, možemo da krenemo od najvećih podskupova. Pošto je brute-force algoritam, možemo i redom da ispitujemo, bez tog poboljšanja.

Algorithm 2 BruteForce_MaxIndependentSet(G)

```
1: Input: Graf G = (V, E)
2: n ← |V|
3: best_set ← ∅
4: best_size ← 0
5: for each S ⊆ V do
6:     if isIndependent(G, S) = true then
7:         if |S| > best_size then
8:             best_size ← |S|
9:             best_set ← S
10:        end if
11:    end if
12: end for
        return best_set
```

(d) Ako problem pripada NP klasi, trebalo bi da poseduje polinomijalnu prove- rljivost, odnosno u slučaju postojanja sertifikata (na primer u ovom slučaju to bi bila klika od k čvorova), trebalo bi da možemo da proverimo ispravnost sertifikata u polinomijalnom vremenu (da proverimo da li je dati skup od k čvorova dat kao klika zaista klika). U kvadratnom vremenu možemo da proverimo da li su svi čvorovi iz potencijalne klike medjusobno povezani. Tako da je ovaj uslov ispunjen.

Dalje, da bismo proverili da li je problem NP težak potrebno je da uzmemo neki za koji znamo već da je NP težak i da ga u polinomijalnom vremenu svedemo na naš posmatrani. U ovom zadatku već nam je rešeno da treba svesti pokrivač čvorova.

Kada bi u grafu G postojao pokrivač veličine l (zovimo ga P) to bi značilo da imamo skup od l čvorova sa kojima su svi ostali povezani ivicama. Odnosno, ne postoji grana u G koja ima oba svoja čvora u $V \setminus P$. Ako posmatramo komplement grafa G, G', onako kako smo ga

na početku definisali, možemo da primetimo da su sve grane izmedju čvorova u $V \setminus P$ prisutne, odnosno imamo kliku veličine $n - l$ gde je n ukupan broj čvorova u V. Pitanje da li u grafu G imamo pokrivač veličine l svodi se na pitanje da li u grafu G' imamo kliku veličine $n - l$.

2. Problem zbira podskupova glasi: Dato je n nenegativnih celih brojeva i ciljna suma k, da li postoji podskup ovih brojeva takav da je njihova suma jednaka k?

   (a) **(2 poena)** Problem particija glasi: Da li se dati skup brojeva S može podeliti na dva podskupa A i $\overline{A} = S - A$ tako da je zbir elemenata u A jednak zbiru elemenata u $\overline{A}$? Objasniti kako se problem zbira podskupova može svesti na problem particija.

   (b) **(2 poena)** Pokazati da problem particija ima rešenje akko problem zbira podskupova ima rešenje.

   (c) **(2 poena)** Objasniti neki aproksimacioni algoritam za problem zbira podskupova.

   (d) **(4 poena)** Odrediti aproksimacioni faktor za algoritam pod c, i napisati njegov pseudokod.

   REŠENJE:

   (a) Neka je S skup svih nenegativnih celih brojeva koje posmatramo i neka je M suma svih elemenata u skupu S. Dodaćemo element $M - 2 * k$ u skup S i tako dobiti novi skup S'. Pokazaćemo da S' ima podelu na dva jednaka dela (po zbiru elemenata) akko u originalnom skupu S postoji podskup brojeva čija je suma jednaka k.

   (b) Pretpostavimo da se S' može podeliti na dva dela tako da su sume tih delova jednake. Element $M - 2 * k$ mora biti u jednoj od te dve polovine, na primer u T'. Suma elemenata u T' mora biti jednaka $1/2 * (M + M - 2 * k)$ (M je bila suma svih elemenata u S pa je suma svih elemenata u S' jednaka $M + M - 2 * k$). Odatle sledi da je suma elemenata u T' jednaka $M - k$. Izbacivanjem elementa $M - 2 * k$ iz T' dobijamo elemente iz originalnog skupa S čija je suma jednaka $M - k - (M - 2 * k) = k$.

   Pretpostavimo da S ima podskup elemenata (zovimo njihov skup T) čija je suma jednaka k. Onda je suma preostalih elemenata jednaka $M - k$. Kada tim elementima u T priključimo element $M - 2 * k$, dobijamo skup T' čija je suma elemenata jednaka $M - 2 * k + k = M - k$. Ovim smo skup S' čija je suma $M + M - 2 * k = 2 * (M - k)$ podelili na dve polovine sa jednakim sumama.

   (c) Neka je $S = \{a_1, a_2, ..., a_n\}$. Jedna mogućnost za aproksimativni algoritam je na primer da sortiramo sve elemente po veličini opadajuće, a potom prolazimo redom od najvećeg ka najmanjem. Potencijalno

rešenje postavimo na 0 na početku i na njega dodajemo elemente redom koji mogu da stanu.

(d) U nastavku je pseudokod:

Require: Niz $\{a_1, a_2, \ldots, a_n\}$ nenegativnih brojeva; kapacitet (ciljna suma) k.

Ensure: Podskup $R \subseteq \{a_1, \ldots, a_n\}$ čiji je zbir $\leq k$

```
1: Sortiraj niz a u opadajućem poretku tako da a_1 ≥ a_2 ≥ ··· ≥ a_n.
2: zbir ← 0                          // Trenutna suma izabranih elemenata
3: R ← ∅                                                  // Izabrani elementi
4: for i ← 1 to n do
5:     if zbir + a_i ≤ k then
6:         zbir ← zbir + a_i
7:         R ← R ∪ {a_i}
8:     end if
9: end for
       return R
```

Dva slučaja su mogla da nastupe. Jedan je da je $R = S$ i u tom slučaju se u promenljivoj *zbir* nalazi suma svih elemenata iz S, koja je ili jednaka k ili manja od k što se lako može utvrditi. Drugi, netrivijalan slučaj, jeste da neki element nije mogao da stane u zbir. Neka je a_j prvi takav element. Svi elementi pre njega $a_1, a_2, ..., a_{j-1}$ su stavljeni u potencijalni zbir i veći su od njega. $zbir + a_j > k$ jer da je bilo manje ili jednako k, a_j bi bio dodat u zbir. Element a_j je sigurno manji ili jednak od $k/2$ (jer da nije zbir bi već bio veći od k, jer su elementi stavljeni u zbir veći od a_j). Ako bi i zbir bio manji ili jednak od $k/2$ tada bi bilo ispunjeno da je $zbir + a_j \leq k/2 + k/2 = k$ i mogli bismo da dodamo a_j u zbir. Dakle, *zbir* već zadovoljava nejednakost $zbir \geq k/2$. S obzirom da ćemo potencijalno dodati neke nenegativne elemente, zbir može samo da se poveća, te će ovakva nejednakost nastaviti da važi. Mi pokušavamo da maksimizujemo zbir brojeva, a da on ne predje vrednost k. Dakle, ovo je maksimizacioni tip aproksimativnog algoritma (pročitati donji tekst), te je faktor jednak 2 (jer je $k/zbir \leq 2$).

Aproksimacioni algoritmi i aproksimacioni faktor: Aproksimacioni algoritam garantovano ima polinomijalno vreme izvršavanja, ali nam ne garantuje da ćemo dobiti najbolje rešenje. Pretpostavimo da radimo sa algoritmom gde svako potencijalno rešenje ima neku cenu, i želimo da se što više približimo optimalnoj ceni. U zavisnosti od tipa problemo, možemo da tražimo minimalnu ili maksimalnu optimalnu cenu. Ako algoritam postiže aproksimacioni faktor $P(n)$ zovemo ga $P(n)$-aproksimacioni algoritam. Za maksimizacione probleme gde je $0 < C < C*$ i C cena nadjenog rešenja, a $C*$ cena optimalnog rešenja, odnos $C*/C$ je aproksimacioni faktor. Za minimizacione probleme gde je $0 < C* < C$ odnos $C/C*$ je aproksimacioni faktor.

3. Pogadjajući skup: Neka je C kolekcija skupova. H je pogadjajući skup za C ako ima neprazan presek sa svakim skupom iz kolekcije C. Pogadjajući skup H je minimalan ako bi uklanjanjem bilo kog elementa iz njega, H prestao da bude pogadjajući. Pogadjajući skup najmanje kardinalnosti je najmanji pogadjajući skup. Problem najmanjeg pogadjajućeg skupa glasi: Neka je data kolekcija skupova C, odrediti najmanji pogadjajući podskup za C.

   (a) **(2 poena)** Za skup $C = \{\{1,4,5\},\{1,2,3\},\{2,6,7\},\{1,3,7\}\}$ dati primer jednog minimalnog pogadjajućeg skupa i jednog najmanjeg pogadjajućeg skupa.

   (b) **(3 poena)** Dati naivan približan algoritam i oceniti njegov faktor aproksimacije u slučaju da su svi skupovi u kolekciji C kardinalnosti 3. Osmisliti primer u kome ovaj algoritam daje pogadjajući skup koji nije najmanji.

   (c) **(3 poena)** Dati jedan pohlepan i jedan randomizovan aproksimativni algoritam za ovaj problem.

   (d) **(2 poena)** Problem pokrivanja skupova glasi: Data je kolekcija skupova S, izabrati najmanji mogući broj skupova iz ove kolekcije tako da se svaki element svakog skupa iz S nalazi u nekom od odabranih skupova. Pokazati da su problem pokrivanja skupova i problem najmanjeg pogadjajućeg skupa suštinski isti.

   REŠENJE:

   (a) Primer jednog minimalnog pogadjajućeg skupa je $H = \{4,6,3\}$. Ako bismo uklonili 4 skup $\{1,4,5\}$ ne bi bio pogodjen. Ako bismo uklonili 3 skup $\{1,3,7\}$ ne bi bio pogodjen. Ako bismo uklonili 6 skup $\{2,6,7\}$ ne bi bio pogodjen. Najmanji pogadjajući skup je $\{1,7\}$. Iz ovog vidimo da nije svaki minimalan pogadjajući skup najmanji. Ali svaki najmanji jeste minimalan. Kada ne bi bio, mogli bi da izbacimo neke elemente i dobijemo manji pogadjajući skup, što je kontradikcija sa tim da je najmanji.

   (b) Neka je S broj različitih elemenata u uniji ponudjenih skupova. Možemo da krenemo redom po skupovima i za svaki koji nije pokriven izabranim elementima, dodamo jedan element iz tog skupa u izabrane elemente. U najgorem slučaju, svi izabrani elementi su iz S, tako da je njihov broj ograničen sa kardinalnošću od S. Sa druge strane, optimalno rešenje ne može da ima manje od $|S|/3$ elemenata, jer svaki element optimalnog rešenja može da bude jednak jednom od tri elementa u podskupu (prvom, drugom ili trećem). Tada je aproksimacioni faktor $\frac{|S|}{|S|/3} = 3$.

   Neka je $C = \{\{1,2,3\},\{1,4,5\},\{1,4,7\}\}$. Očigledno je najmanji minimalni pogadjajući skup jednak $\{1\}$. Medjutim, pošto naivni algoritam proizvoljno bira element da pokrije skup koji je po redu, on

će na primer da iz prvog $\{1, 2, 3\}$ izabere 2, iz drugog 4, a iz trećeg neće da bira jer je treći pokriven sa 4.

(c) Pohlepan aproksimativni algoritam: Sortiramo elemente po broju skupova koje pokrivaju. Biramo od najvećeg ka najmanjem sve dok ne pokrijemo sve skupove.
Randomizovan aproksimativni algoritam: Naivni algoritam koji smo imali u prethodnoj tački može se lako pretvoriti u randomizovani, ako na slučajan način biramo elemente kojima želimo da pokrivamo nepokrivene skupove. Takodje, postoji mogućnost složenijeg algoritma ako po nekoj formuli izračunamo verovatnoće odabira pojedinih elemenata i biramo ih u skladu sa tim verovatnoćama.

(d) Neka je U skup elemenata koji se javljaju u skupovima $\mathcal{F} = \{F_1, F_2, ..., F_n\}$ problema pogadjajućeg skupa. Za svaki element iz U definisemo na sledeći način skupove $S_u = \{F_i | u \in F_i\}$. Stavimo da je $S = \{S_u | u \in U\}$ i sada rešavamo problem pokrivanja skupova.
Pokažimo da je $H \subseteq U$ pogadjajući skup za $\mathcal{F}$ ako i samo ako odgovarajuća kolekcija

$$\{S_u \mid u \in H\}$$

pokriva $X = \mathcal{F}$.

(1) Ako je H pogadjajući skup:
Za svaki $F \in \mathcal{F}$ postoji $u \in H$ takav da je $u \in F$. Ali to znači da je

$$F \in S_u \quad \text{za taj } u,$$

gdje je

$$S_u = \{F \in \mathcal{F} \mid u \in F\}.$$

Stoga, unija svih takvih S_u za $u \in H$ pokriva čitavo $\mathcal{F}$:

$$\bigcup_{u \in H} S_u = \mathcal{F}.$$

Dakle, izbor

$$\{S_u \mid u \in H\}$$

predstavlja pokrivanje skupa $\mathcal{F}$.

(2) Ako $\{S_u \mid u \in H\}$ pokriva $\mathcal{F}$:
Za svaki $F \in \mathcal{F}$ postoji barem jedan $u \in H$ takav da

$$F \in S_u.$$

Po definiciji skupa S_u, to znači da je $u \in F$. Time se dobija da

$$H \cap F \neq \varnothing \quad \text{za svaki } F \in \mathcal{F},$$

tj. H je pogadjajući skup za $\mathcal{F}$. Ova dvosmerna implikacija pokazuje da pronalaženje najmanjeg pogadjajućeg skupa u instanci $(U, \mathcal{F})$ odgovara pronalaženju najmanjeg pokrivanja skupa u instanci $(X, \mathcal{S})$.