Άσκ3 (Σύνολο κρούσης μέσω linear programming) Βλάχος Δημήτριος

Δεδομένα

> Σύνολο με n αντικείμενα $A = \{a_1, a_2, \dots, a_n\}$.
> Οικογένεια $B = \{B_1, B_2, \dots, B_k\}$ υποσυνόλων του A.
> $\forall$ αντικείμενο $a_i \in A$ έχει $c_i > 0$ (θετικό κόστος)
> Ένα υποσύνολο $H \subseteq A$ αποτελεί 1 σύνολο κρούσης της B αν για $\forall$ $B_i \in B$ η τομή $(B_i \cap H)$ δεν είναι κενή
(Δηλ: το H περιέχει τουλ. 1 στοιχείο από $\forall$ υποσύνολο B_i της B)
> Το κόστος του H = το sum των κόστους $\forall$ στοιχείων του.
$\Longleftrightarrow$ $\boxed{C(H) = \sum_{a_i \in H} c_i}$
> Problem Συνόλου κρούσης: Ποιο το ελάχιστο κόστος για την οικογένεια B;

(α) Ν.Δ.Ο. το Σύνολο κρούσης είναι γενίκευση του Κομβικού Καλύμματος

Λύση: > Ας ορίσουμε τα δύο προβλήματα με βάση τα δεδομένα της άσκησης & τη θεωρία μας έτσι ώστε να μπορέσουμε να καταλήξουμε στο ζητούμενο συμπέρασμα.

> Κομβικό κάλυμμα: Έστω γράφος $G=(V,E)$ με $|V|$ κόμβους, $|E|$ ακμές.
Έστω επίσης υποσύνολο $U \subseteq V$ είναι κομβικό κάλυμμα ανν για $\forall$ edge $(u,v) \in E$ ισχύει $u \in U$ ή $v \in U$ ή $u, v \in U$.
Τα βάρη των κόμβων είναι $w_i \geq 0$ για $\forall$ $i \in V$.
Στόχος: να βρούμε ένα κομβικό κάλυμμα, έστω U με το ελάχιστο συνολικό βάρος. $\Longleftrightarrow$ $\boxed{\min \sum_{i \in U} w_i}$

> Σύνολο κρούσης: Έστω ένα σύνολο $A = \{a_1, a_2 \dots a_n\}$ & μια οικογένεια υποσυνόλων $B = \{B_1, B_2, \dots, B_k\}$ τ.ω. $B_i \subseteq A$
Σύνολο κρούσης είναι 1 subset $H \subseteq A$ τ.ω. $\boxed{B_i \cap H \neq 0}$ για $\forall$ i.
$\forall$ στοιχείο έχει κόστος $c_i > 0$.
Στόχος: Να βρούμε 1 υποσύνολο (σύνολο κρούσης) H τ.ω.
$\boxed{\min \sum_{a_i \in H} c_i}$

> Θα προσπαθήσουμε να δείξουμε ότι το απλούστερο πρόβλημα (Σύνολο Κρούσης) είναι γενίκευση του συνθετότερου (Κομβικό κάλυμμα). Μετασχηματίζοντας το απλούστερο & ταιριάζοντας τα επιμέρους δεδομένα των 2 προβλημάτων.

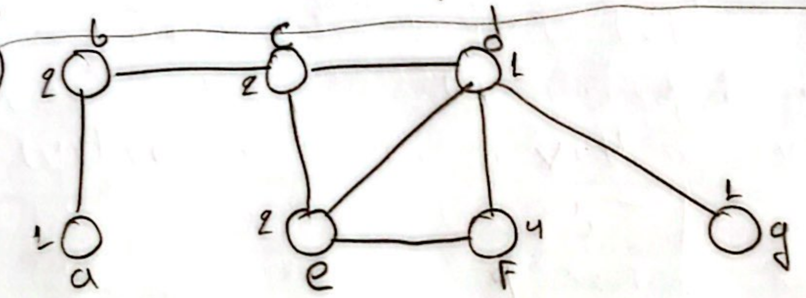
Άσκ3) συνέχεια (α)

- **Βάρη ⟺ κόστη:**
  Βάζουμε τα <u>Βάρη</u> από το Κομβικό κάλυμμα <u>ως</u> <u>κόστη</u> των Συνόλων κρούσης.
  Συνεπώς για $\forall i$: $\underline{c_i = w_i}$

- **Κόμβοι ⟺ αντικείμενα:**
  A : το σύνολο αντικειμένων $\{a_1, a_2 \ldots a_n\}$ του συνόλου κρούσης
  V : το σύνολο κόμβων του Κομβικού Καλύμματος.
  Συνεπώς $\underline{A = V}$

- **Ακμές ⟺ Υποσύνολα:**
  Για $\forall$ ακμή $(u,v) \in E$ του κομβικού καλύμματος αντιστοιχούμε ένα υποσύνολο $B_i \in B$ του Συνόλου Κρούσης.
  Συνεπώς: $B_i = (u,v)$ για $\forall i$

ΠΧ



$V = \{a, b, c, d, e, F, g\}$, $E = \{(a,b), (b,c), (c,d), (c,e), (e,d), (e,f), (d,f), (d,g)\}$

$A = \{a, b, c, d, e, f, g\}$

$B_1 = (a,b)$, $B_2 = (b,c)$, $B_3 = (c,d)$, $B_4 = (c,e)$, $B_5 = (e,d)$
$B_6 = (e,f)$, $B_7 = (d,f)$, $B_8 = (d,g)$

$c_a = 1$, $c_b = 2$, $c_c = 2$, $c_d = 1$, $c_e = 2$, $c_f = 4$, $c_g = 1$

Χρήση του G(V,E) από διαφάνειες (last page)

Ασκ3 (β) Περιγράψτε ένα "Ακέραιο Πρόγραμμα" που θα περιγράφει τη βέλτιστη λύση του "Συνόλου Κρούσης".
Υπόδειξη: Ορίστε μια μεταβλητή $X_i \in \{0,1\}$ για $\forall$ αντικείμενο $a_i \in A$

Λύση :> Ορίζουμε την μεταβλητή: $X_i \in \{0,1\}$ για $\forall$ αντικείμενο $a_i \in A$

$$\begin{cases} X_i = 1, & \text{Αν } a_i \text{ υπάρχει στο Σύνολο Κρούσης} \\ X_i = 0, & \text{Αν } a_i \text{ δεν υπάρχει } \text{»}. \end{cases}$$

> Ορίζουμε τη συνάρτηση (αντικειμενική):

$$\boxed{\min \sum_{j=1}^{n} c_j x_j}$$

το κόστος να $\exists$ το a_j στο Σύνολο Κρούσης

> Ορίζουμε τους περιορισμούς:

Για $\forall$ υπο-σετ (υποσύνολο) $B_i \in B$ $\sum_{a_j \in B_j} x_j \geq 1 \quad \forall i = 1, 2, \ldots k$

Έτσι εξασφαλίζουμε ότι τουλ. 1 στοιχείο από το B_i θα $\exists$ στο Σύνολο Κρούσης.

Δηλ. το Σύνολο Κρούσης $\cap B_i \neq \emptyset$ για $\forall i$.

Πχ

$$\begin{cases} \min\{x_1 + 3x_2 + 2x_3 + 4x_4\} \\ x_1 + x_2 \geq 1 \\ x_2 + x_3 \geq 1 \\ x_3 + x_4 \geq 1 \\ x_1, x_2, x_3, x_4 \in \{0,1\} \end{cases}$$

Ψάχνουμε το βέλτιστο "Σύνολο Κρούσης" που έχει τομή $\neq 0$ με τα υποσύνολα $\{a_1, a_2\}$, $\{a_2, a_3\}$, $\{a_3, a_4\}$

Άσκ3 (γ) > Έστω ότι ∀ σύνολο $B_i \in B$ έχει size $|B_i| \le 6$.

> Περιγράψτε πως μπορούμε να υπολογίσουμε μια b-προσεγγιστική λύση του Συνόλου Κάλυψης, μέσω της τεχνικής Χαλάρωσης του Ακέραιου Προγράμματος του (B) σε Γραμμικό Πρόγραμμα

Λύση > Θα προσπαθήσουμε να "χαλαρώσουμε" τους ακέραιους περιορισμούς

> • $\boxed{\min \sum_{i=1}^{n} c_i x_i}$

• $\sum_{a_i \in B_j} x_i \ge 1 \quad \forall j$, • $x_i \in \{0,1\} \quad \forall i$ } Δεδομένα

> $x_i \in \{0,1\} \implies x_i \in [0,1]$ Δηλ: Πάμε σε συνεχές πεδίο τιμών.

> • $\boxed{\min \sum_{i=1}^{n} c_i x_i}$

• $\sum_{a_i \in B_j} x_i \ge 1 \quad \forall j$, • $x_i \in [0,1] \quad \forall i$

> Αλγόριθμος:

```
• ΣΚ = ∅; // Αρχικοποίησε με κενό Σύνολο Κάλυψης
• For(∀ ai∈A){ // Για ∀ στοιχείο του A
    num = rand([0,1]); // generate a random num ∈ [0,1]
    if(num ≤ (*)) {
        ΣΚ ← ΣΚ ∪ ai; // συμπεριέλαβε το element ai
    }
  }
```

(*) Εδώ δεν είμαι σίγουρος τι πρέπει να μπει
Ίσως 1/6

Ασκ1

Δεδομένα

> Έστω $G=(V,E)$ simple graph

> $M \subseteq E$ του G αποτελεί ταίριασμα αν στο $G_M(V,M)$ (sub-graph) του G ∀ κορυφή έχει degree $= 0$ ή $= 1$.

⟺ ∀ κορυφή έχει max 1 γείτονα στο G_M.

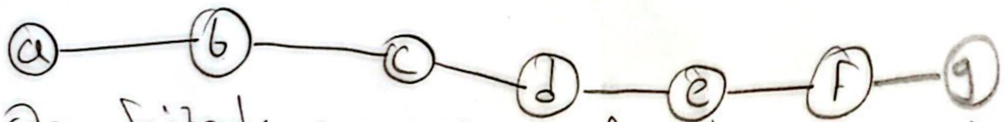
> Το ταίριασμα M είναι μέγιστο αν για ∀ άλλο ταίριασμα M' του G ισχύει $|M| \geq |M'|$.

> Ο υπολογισμός ενός μέγιστου ταιριάσματος μπορεί να γίνει σε πολυωνυμικό χρόνο, αλλά ⟹ περίπλοκοι Αλγόριθμοι στη γενική περίπτωση.

> Απλοϊκός αλγόριθμος:

```
M = Ø;  // Ταίριασμα αρχικά empty
For(e = {x,y} ∈ E){  // Για ∀ ακμή του γράφου
   if (οι nodes x & y ελεύθερες){  // Check if M∪e ταίριασμα
      M = M∪e;
   }
}
```

(α.) Δώστε ΠΧ που ο παραπάνω Αλγόριθμος υπολογίζει ταίριασμα που δεν είναι μέγιστο

> 

> Θα δείξουμε πως τρέχει ο αλγόριθμος για το γράφο

> $\{a,b\}$: a & b είναι ελεύθερες κορυφές.

$\Longrightarrow M = M \cup \{a,b\} = \{\{a,b\}\}$

> $\{b,c\}$: b όχι ελεύθερη, c ελεύθερη

$\Longrightarrow$ M ως έχει

> $\{c,d\}$: c & d είναι ελεύθερες

$\Longrightarrow M = M \cup \{c,d\} = \{\{a,b\},\{c,d\}\}$

> $\{d,e\}$: d όχι ελεύθερη, e ελεύθερη

$\Longrightarrow$ M ως έχει

> $\{e,f\}$: e & f ελεύθερες.

$\Longrightarrow M = M \cup \{e,f\} = \{\{a,b\},\{c,d\},\{e,f\}\}$

> $\{f,g\}$: $\Longrightarrow$ M ως έχει

> **Αποτέλεσμα** $M = \{\{a,b\},\{c,d\},\{e,f\}\}$

Είναι max; $\Longrightarrow$ **ΟΧΙ** απαραίτητα!

1

Ασκ 1 (α) <u>συνέχεια</u>

Αν οι ακμές επιλεχθούν με διαφορετική σειρά, θα καταφέρουμε ίσως να έχουμε μικρότερο ταίριασμα.

> $\{f,g\}$: f, g ελεύθερες κορυφές

$$\Longrightarrow M = M \cup \{f,g\} = \{\{f,g\}\}$$

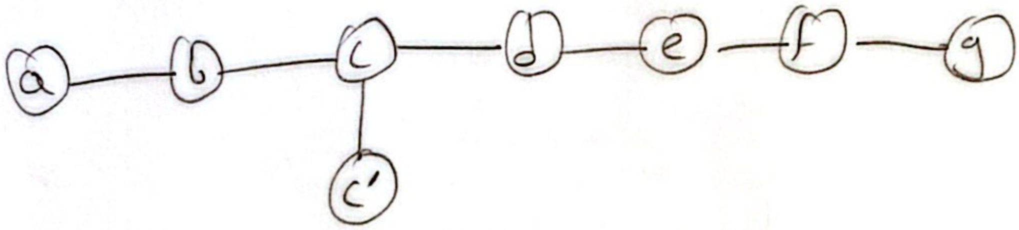
> $\{a,b\}$: a, b ελεύθερες κορυφές

$$\Longrightarrow M = M \cup \{a,b\} = \{\{f,g\},\{a,b\}\}$$

> $\{d,e\}$: d, e ελεύθερες κορυφές

$$\Longrightarrow M = M \cup \{d,e\} = \{\{f,g\},\{a,b\},\{d,e\}\}$$

Τώρα να προσθέσουμε μια ακόμα κορυφή στο γράφο:



> $\{c,c'\}$: c, c' ελεύθερες κορυφές

$$\Longrightarrow M = M \cup \{c,c'\} = \{\{f,g\},\{a,b\},\{d,e\},\{c,c'\}\}$$

> Ενώ το M στο προηγούμενο του αλγορίθμου δεν θα άλλαζε. Άρα μπορεί ο αλγόριθμος να επιστρέψει M που δεν είναι μέγιστο.

Άσκ1 (β) Ν.Δ.Ο ο αλγόριθμος είναι 2-προσεγγιστικός
Δηλ: υπολογίζει ταίρι M : $|M| \geq |M^*|/2$, M^*: μέγιστο στο G

Λύση

> Πρέπει να δείξω ότι: $|M| \geq |M^*|/2 \iff$

[Το ταίριασμα M έχει μέγεθος τουλ. το μισό του μεγέθους του μέγιστου ταιριάσματος M^*]

> Έστω λοιπόν το M είναι το ταίριασμα που θα υπολογίσει ο αλγόριθμος.

> Το πώς το κάνει το είδαμε & το ερώτημα (α).

> Προσθέτει ακμές ανν. & οι 2 κόμβοι της είναι ελεύθεροι.

> Αν $M \in M \cup \{u, v\}$ // οι κόμβοι είναι ελεύθεροι
> Πλέον u, v δεν είναι ελεύθεροι!

> Αν $|M| = n \Longrightarrow$ το # των κόμβων από το ταίριασμα είναι $2*n$

> Έστω τώρα το max ταίριασμα M^*.
> $\forall$ ακμή "πιάνει" 2 κόμβους.

Όσον αφορά το M^* δεν είναι δυνατό να έχει & τους 2 κόμβους όμοιους με μια ακμή του M

Ασκ 2 > KNAPSACK (OPT) (Πρόβλημα του σακιδίου)

> W (>0, integer) αντιστοιχεί στη χωρητικότητα του σακιδίου

Δεδομένα

> n αντικείμενα (1, 2, ..., n) με ∀ αντικείμενο i : w_i & v_i (≥ 0 int) αξία.

> **Ζητούμενο**: Επιλογή ενός συνόλου αντικειμένων $S \subseteq \{1, 2, \dots, n\}$ που να μεγιστοποιεί το sum $\boxed{\sum_{i \in S} v_i}$ με τον περιορισμό $\boxed{\sum_{i \in S} w_i \leq W}$ ⟺ Να επιλέξουν αντικείμενα με την μέγιστη δυνατή αξία, ώστε το συνολικό βάρος να μην υπερβαίνει τη χωρητικότητα του σακιδίου.

> Σχεδιασμός άπληστου $\boxed{\text{Αλγορίθμου}}$:

> Σε ∀ step επιλέγω $\boxed{\text{αντικείμενο}}$ που η αξία ανά μονάδα βάρους είναι μέγιστη ανάμεσα στα αντικείμενα που μπορούν να προστεθούν στο $\boxed{\text{σακίδιο}}$ χωρίς να υπερβαίνει την αντοχή του χωρητικότητα.

> Υλοποίηση: τα $\boxed{\text{αντικείμενα}}$ είναι ταξινομημένα σε φθίνουσα τάξη ως προς το λόγο v_i/w_i.

> Εξετάζουμε με σειρά τα αντικείμενα. Εισάγουμε 1 αντικείμενο στο σακίδιο S αν το βάρος δεν υπερβαίνει την υπολειπόμενη χωρητικότητα C.

> $\boxed{\text{Αλγόριθμος}}$:

Έστω $v_1/w_1 \geq v_2/w_2 \geq \dots \geq v_n/w_n$, v_i = αξία, w_i = βάρος (i-οστό element)

```
C = W; S = ∅; // Αρχικοποίηση
For(i=1 to n){
    if(wi ≤ C){
        S = S∪{i}; C = C-wi;
    }
}
```

(α.) Ν.Δ.Ο ο αλγόριθμος Greedy-Knapsack δεν είναι ρ-προσεγγιστικός για το πρόβλημα KNAPSACK(OPT) για ρ > 1

Λύση: > Πρέπει: $\boxed{\dfrac{\text{wy\_greedy solution}}{\text{optimal solution}} \geq \dfrac{1}{\rho}}$

> Η λύση που παρέχει ο greedy αλγόριθμος θα πρέπει να είναι αυθαίρετα ως προς το χειρότερο συγκριτικά με τον optimal αλγόριθμο.

Ασκ 2 (α) ανίχνια

> Έστω element 1: με $\frac{v_1}{w_1} = \frac{1000}{500} = 2$

Έστω element 2: με $\frac{v_2}{w_2} = \frac{999}{1} = 999$

Προφανώς $999 > 2 \Rightarrow$ Επιλογή element 2 από τον greedy Αλγόριθμο

Άρα: Greedy λύση $= 999$

Όμως η βέλτιστη είναι η επιλογή του element 1.

> $$\boxed{\frac{\text{my\_greedy solution}}{\text{optimal solution}} = 0.99}$$

> Για να δείξουμε ότι ο greedy δεν είναι ρ-προσεγγιστικός για $\rho > 1$ πρέπει να δείξουμε ότι το κλάσμα $\to 0$ (μπορεί)

Δηλ: $\frac{\text{my\_greedy solution}}{\text{optimal solution}} \to 0$

Έστω element 1: με $\frac{v_1}{w_1} = \frac{n^2}{n} = n$

Έστω element 2: με $\frac{v_2}{w_2} = \frac{n^2+ct}{ct}$, ct: μικρή ακέραια σταθερά

> $$\frac{n^2+ct}{n^2} = \boxed{1 + \frac{ct}{n^2}}$$

Για $n \to \infty$ η ποσότητα $\to 1$

(Προφανώς δεν θα γίνει ποτέ $= 1$)

[Συνεπώς δείξαμε ότι το $\frac{\text{my\_greedy solution}}{\text{optimal solution}}$ κλάσμα μπορεί να γίνει όσο μικρό θέλουμε]

> Συνεπώς ο greedy αλγόριθμος για input element που έχουν τεράστιες διαφορές στις αναλογίες $\frac{\text{τιμή}_i}{\text{βάρος}_i}$ δεν είναι ρ-προσεγγιστικός ▨

2

Scanned with CamScanner

Ασκ 2 (β) Το β-KNAPSACK(OPT), $\beta \in [0,1]$ είναι η ειδική case του KNAPSACK(OPT) στην οποία για $\forall$ element i : $w_i \leq \beta W$.

Ν.Δ.Ο.: Ο greedy αλγόριθμος Greedy-Knapsack είναι ένας $\left(\frac{1}{1-\beta}\right)$-προσεγγιστικός αλγόριθμος για το πρόβλημα β-KNAPSACK(OPT)

Λύση: > Πρέπει: Η my_greedy solution να είναι τουλ. $\frac{1}{1-\beta}$ φορές η optimal solution.

$$\Longleftrightarrow \frac{\text{my\_greedy solution}}{\text{optimal solution}} \geq \frac{1}{1-\beta} \Longleftrightarrow$$

$$\text{optimal solution} \geq (1-\beta) * \text{my\_greedy solution}$$

Αυτό προφανώς πρέπει να το αποδείξουμε.

> Σ_g : το σύνολο των elements που είναι το input του greedy αλγορίθμου

Σ_o : το σύνολο των elements που είναι το input του optimal.

V_g : οι τιμές των στοιχείων Σ_g

V_o : οι τιμές των στοιχείων Σ_o.

> Τα elements επιλέγονται με βάση το κλάσμα $\frac{v_i}{w_i}$.

|elements που επιλέγονται| <= W

> Έστω m-οστό το 1ο αντικείμενο που δεν χωράει στο σακίδιο. Συνεπώς δεν επιλέγεται.

Για όλα τα αντικείμενα που έχουν επιλεχθεί ισχύει:

$v_i / w_i \geq v_m / w_m$ για $i = 1, 2, \ldots, m-1$

> Προφανώς $V_o < V_g$ } Διότι το sum των τιμών που επιλέγει ο greedy υπερβαίνει το sum των τιμών που επιλέγει ο optimal.

> $\Longrightarrow V_o <= V_g + V_m$ } Διότι: το sum των τιμών που επιλέγει ο optimal στην ορισμένη περίπτωση είναι = με το sum των τιμών του greedy μαζί με την αξία του m-οστού element.

> Για το v_m:

$v_m \leq W * \beta$ } Γιατί το m-οστό element δεν μπορεί να επιλεχθεί εξαιτίας του βάρους του W_m.

$\boxed{\text{Άσκ 2}}$ (β) > Άρα: $\boxed{V_o \le V_g + u_m \le V_g + w \cdot b \, \frac{u_1}{w_1}}$ (1)

η μεγαλύτερη αναλογία τιμής / βάρους που επιλέγει από τον greedy

> Επειδή u_1/w_1 η μεγαλύτερη αναλογία που επιλέγει από τον greedy: $\boxed{u_m \le V_g * b}$ (2)

> Από (1) & (2) προκύπτει:

$$\boxed{\underbrace{V_o \le V_g + V_g * b}\le V_g + w \cdot b \cdot u_1/w_1}$$

$\Longleftrightarrow V_o \le V_g(1+b) \Longleftrightarrow \frac{V_o}{V_g} \le (1+b)$

$\Longleftrightarrow \boxed{\frac{V_g}{V_o} \ge \frac{1}{1+b}}$

> Συνεπώς όπως ο greedy αλγόριθμος είναι $\left(\frac{1}{1+b}\right)$ προσεγγιστικός για το problem b-KNAPSACK(OPT)