ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΠΡΟΗΓΜΕΝΑ ΘΕΜΑΤΑ ΑΛΓΟΡΙΘΜΩΝ

(2020-2021)

1η Σειρά Ασκήσεων

Ονοματεπώνυμο:

- Χρήστος Τσούφης

Αριθμός Μητρώου:

- 03117176

Στοιχεία Επικοινωνίας:

- el17176@mail.ntua.gr

1η Άσκηση

Δίνεται το Γραμμικό Πρόγραμμα ($Π_1$):

minimize: $3x_1 + x_2 + x_3 + x_4$

subject to: $3x_1 - x_2 - 7x_3 + x_4 = -3$

$3x_1 - 4x_3 + x_4 = -1$

$x_3 \leq 3$

$x_1, x_2, x_3, x_4 \geq 0$

1. *Αρχικά, ζητείται ο σχεδιασμός του συνόλου των εφικτών λύσεων του ($Π_1$) καθώς και οι κορυφές του. Έπειτα ζητείται ο σχεδιασμός της κατεύθυνσης βελτιστοποίησης και η εύρεση της βέλτιστης λύσης.*

   Θέτοντας $x = 3x_1 + x_4$, $y = x_2$, $z = x_3$, προκύπτει ένα ισοδύναμο γραμμικό πρόβλημα ($Π_2$) ως εξής:

   minimize: $x + y + z$
   subject to: $x - y - 7z = -3$
   $x - 4z = -1$
   $z \leq 3$
   $x, y, z \geq 0$

   Από τα παραπάνω, προκύπτει ότι το x μπορεί να απαλειφθεί αφού $x - 4z = -1$. Επιπλέον, επειδή πρέπει και $x \geq 0$, προκύπτει ότι $4z - 1 \geq 0 \Rightarrow z \geq \frac{1}{4}$. Επομένως, η νέα αντικειμενική συνάρτηση, γίνεται $5z + y - 1$. Έτσι, προκύπτει το νέο ισοδύναμο γραμμικό πρόβλημα ($Π_3$):

   minimize: $y + 5z$
   subject to: $y + 3z = 2$
   $z \leq 3$
   $y \geq 0$
   $z \geq \frac{1}{4}$

   Οπότε, $OPT_{Π_1} = OPT_{Π_2} = OPT_{Π_3} - 1$.

   Οι περιορισμοί είναι οι εξής:
   - Η ευθεία $ε = \{(z, y) \in \mathbb{R}^2 : y + 3z = 2\}$
   - Το ημι-επίπεδο $π_1 = \{(z, y) \in \mathbb{R}^2 : z \leq 3, y \geq 0, z \geq \frac{1}{4}\}$

   Άρα, το σύνολο των εφικτών λύσεων θα είναι το η ευθεία ε που ανήκει στο ημι-επίπεδο $π_1$. Η βέλτιστη λύση του $Π_3$ είναι η $(z, y) = (\frac{1}{4}, \frac{5}{4})$. Συνεπώς, το $Π_1$ θα έχει βέλτιστη λύση, $x = [0, \frac{5}{4}, \frac{1}{4}, 0]^T$ και objective value την $\frac{3}{2}$.

2. *Ζητείται η διατύπωση του δυικού προγράμματος ($ΔΠ_1$) του ($Π_1$). Επίσης ζητείται η διατύπωση των complementary slackness συνθηκών για τα ($Π_1$) και ($ΔΠ_1$) και η χρήση τους για τον υπολογισμό της βέλτιστης λύσης του ($ΔΠ_1$).*

   Το δυικό πρόγραμμα δημιουργείται με πολλαπλασιασμό του 3ου περιορισμού με -1 ώστε το $Π_1$ να έχει τη μορφή $Ax \leq b$. Έτσι, προκύπτει:

   maximize: $-3y_1 - y_2 - 3y_3$
   subject to: $3y_1 + 3y_2 \leq 3$
   $-y_1 \leq 1$
   $-7y_1 - 4y_2 - y_3 \leq 1$
   $y_3 \geq 0$

   Οπότε, οι complementary slackness συνθήκες για 2 βέλτιστες λύσεις των ($ΔΠ_1$) και ($Π_1$) θα είναι:

   a) $x_1*(3y_1 + 3y_2 - 3) = 0$
   b) $x_2*(y_1+1) = 0$
   c) $x_3*(7y_1 + 4y_2 + y_3 + 1) = 0$
   d) $x_4*y_3 = 0$

   e) $y_1*(3x_1 + x_2 - 7x_3 + x_4 + 3) = 0$
   f) $y_2*(3x_1 - 4x_3 + x_4 + 1) = 0$
   g) $y_3*(x_3 - 3) = 0$

   Η βέλτιστη λύση για το $ΔΠ_1$ προκύπτει από το συνδυασμό της βέλτιστης λύσης του $Π_1$ και των complementary slackness συνθηκών. Έτσι, θα ισχύει:

   $x_2 \neq 0 \overset{b}{\Rightarrow} y_1 = -1$

   $x_3 \neq 3 \overset{g}{\Rightarrow} y_3 = 0$

   $x_3 \neq 0 \overset{c}{\Rightarrow} 7y_1 + 4y_2 + y_3 + 1 = 0 \overset{\substack{y_3=0 \\ y_1=-1}}{\Longrightarrow} y_2 = \frac{3}{2}$

   Οπότε, μια βέλτιστη λύση του $ΔΠ_1$ είναι η y $[-1, \frac{3}{2}, 0]^T$ και η objective value είναι:

   $-3*(-1) - \frac{3}{2} = \frac{3}{2}$, όπως ήταν αναμενόμενο.

2η Άσκηση

Δίνεται το Γραμμικό Πρόγραμμα ($Π_2$):

minimize: $-10x_1 + 57x_2 + 9x_3 + 24x_4$

subject to: $0.5x_1 - 5.5x_2 - 2.5x_3 + 9x_4 + x5 = 0$

$0.5x_1 - 1.5x_2 - 0.5x_3 + x_4 + x_6 = 0$

$x_1 + x_7 = 1$

$x_1, x_2, x_3, x_4, x_5, x_6, x_7 \geq 0$

Ζητείται η λύση (2 φορές) του $Π_2$ με τον αλγόριθμο Simplex, ξεκινώντας με βασικές μεταβλητές τις x_5, x_6, x_7 και ακολουθώντας τους παρακάτω κανόνες για την εναλλαγή στηλών (pivoting) στη βάση (την πρώτη φορά τον ένα, την δεύτερη τον άλλο):

1. *(i) ως νέα βασική μεταβλητή επιλέγεται εκείνη με το ελάχιστο ανηγμένο κόστος, και (ii) αν υπάρχουν δύο ή περισσότερες βασικές μεταβλητές υποψήφιες για έξοδο από τη βάση, επιλέγεται εκείνη με τον ελάχιστο δείκτη.*

2. *(i) ως νέα βασική μεταβλητή επιλέγεται εκείνη (από τις μεταβλητές με αρνητικό ανηγμένο κόστος) με τον ελάχιστο δείκτη, και (ii) αν υπάρχουν δύο ή περισσότερες βασικές μεταβλητές υποψήφιες για έξοδο από τη βάση, επιλέγεται εκείνη με τον ελάχιστο δείκτη.*

Κάθε στήλη αναφέρεται σε μια μεταβλητή του προγράμματος, η οποία δηλώνεται στην κορυφή εκτός, εκτός της τελευταίας στήλης που αναφέρεται στο δεξί μέλος των constraint functions. Επιπλέον, κάθε στήλη αναφέρεται σε ένα constraint, εκτός της τελευταίας που αναφέρεται στην αντικειμενική συνάρτηση. Στα αριστερά δηλώνονται οι βασικές εφικτές μεταβλητές και κάτω δεξιά δηλώνεται το κόστος της μέχρι τώρα λύσης. Με πράσινο χρώμα δηλώνεται η εισερχόμενη βασική μεταβλητή και η αντίστοιχη στήλη περιστροφής και με κόκκινο χρώμα η εξερχόμενη βασική μεταβλητή και η αντίστοιχη γραμμή περιστροφής. Επίσης, περιλαμβάνεται μια επιπλέον στήλη όπου αναφέρεται το θ-ratio κάθε γραμμής. Τέλος, επισημαίνεται πως η σύμβαση θέλει το maximum problem ως standard form ($\min c^T x \rightarrow - \max - c^T x$).

a1) Για την 1η εκτέλεση του αλγορίθμου, παρουσιάζονται τα Simplex Tableu.

vars	**x_1**	**x_2**	**x_3**	**x_4**	**x_5**	**x_6**	**x_7**	**b**	**ratio**
x_5	0.5	-5.5	-2.5	9	1	0	0	0	0
x_6	0.5	-1.5	-0.5	1	0	1	0	0	0
x_7	1	0	0	0	0	0	1	1	1
f	-10	57	9	24	0	0	0	0	**0**

vars	**x_1**	**x_2**	**x_3**	**x_4**	**x_5**	**x_6**	**x_7**	**b**	**ratio**
x_1	1	-11	-5	18	2	0	0	0	-
x_6	0	4	2	-8	-1	1	0	0	0
x_7	0	11	5	-18	-2	0	1	1	1/11
f	0	53	-41	204	20	0	0	0	**0**

vars	x_1	x_2	x_3	x_4	x_5	x_6	x_7	b	ratio
x_1	1	0	0.5	-4	-0.75	2.75	0	0	0
x_2	0	1	0.5	-2	-0.25	0.25	0	0	0
x_7	0	0	-0.5	4	0.75	-2.75	1	1	-
f	0	0	-14.5	90	13.5	13.25	0	0	**0**

vars	x_1	x_2	x_3	x_4	x_5	x_6	x_7	b	ratio
x_3	2	0	1	-8	-1.5	5.5	0	0	-
x_2	-1	1	0	2	0.5	-2.5	0	0	0
x_7	1	0	0	0	0	0	1	1	-
f	29	0	0	-18	-15	93	0	0	0

vars	x_1	x_2	x_3	x_4	x_5	x_6	x_7	b	ratio
x_3	-2	4	1	0	0.5	-4.5	0	0	0
x_4	-0.5	0.5	0	1	0.25	-1.25	0	0	0
x_7	-1	0	0	0	0	0	1	1	-
f	20	9	0	0	-10.5	70.5	0	0	**0**

vars	x_1	x_2	x_3	x_4	x_5	x_6	x_7	b	ratio
x_5	-4	8	2	0	1	-9	0	0	-
x_4	0.5	-1.5	-0.5	1	0	1	0	0	0
x_7	1	0	0	0	0	0	1	1	-
f	-22	93	21	0	0	-24	0	0	**0**

vars	x_1	x_2	x_3	x_4	x_5	x_6	x_7	b	ratio
x_5	0.5	-5.5	-2.5	9	1	0	0	0	0
x_6	0.5	-1.5	-0.5	1	0	1	0	0	0
x_7	1	0	0	0	0	0	1	1	1
f	-10	57	9	24	0	0	0	0	**0**

Παρατήρηση: Δημιουργείται και πάλι η βάση (x_5, x_6, x_7) έχοντας πραγματοποιήσει κύκλο. Αυτό συμβαίνει διότι, οι περιορισμοί του συγκεκριμένου προβλήματος είναι degeneracy. Για αυτό το λόγο, ο αλγόριθμος αποτυγχάνει να επιστρέψει βέλτιστη λύση για το πρόβλημα και η μέθοδος Simplex απαιτεί τροποποιήσεις ώστε να επιστρέψει λύση, η οποία λόγω αυτής της ιδιότητας θα περιέχει και μηδενικές τιμές για βασικές μεταβλητές.

a2) Για την εκτέλεση του αλγορίθμου με τον νέο κανόνα που εισάγεται (κανόνας του Bland), ουσιαστικά αλλάζει η εκτέλεση μόνο στην τελευταία από τις παραπάνω επαναλήψεις, ώστε να τερματίσει. Έτσι,

vars	**x_1**	**x_2**	**x_3**	**x_4**	**x_5**	**x_6**	**x_7**	**b**	**ratio**
x_5	-4	8	2	0	1	-9	0	0	-
x_4	0,5	-1,5	-0,5	1	0	1	0	0	0
x_7	1	0	0	0	0	0	1	1	-
f	-22	93	21	0	0	-24	0	0	**0**

vars	**x_1**	**x_2**	**x_3**	**x_4**	**x_5**	**x_6**	**x_7**	**b**	**ratio**
x_5	01	-20	-2	8	1	-1	0	0	-
x_1	1	-3	-1	2	0	2	0	0	-
x_7	0	3	1	-2	0	-2	1	1	1
f	0	27	-1	44	0	20	0	0	**0**

vars	**x_1**	**x_2**	**x_3**	**x_4**	**x_5**	**x_6**	**x_7**	**b**	**ratio**
x_5	0	-14	0	4	1	-5	2	2	-
x_1	1	0	0	0	0	0	1	1	-
x_3	0	3	1	-2	0	-2	1	1	-
f	0	30	0	42	0	18	1	1	**1**

Παρατήρηση: Εδώ ο αλγόριθμος Simplex τερματίζει καθώς η τελευταία γραμμή έχει αποκλειστικά μη αρνητικές καταχωρήσεις. Συνεπώς, το Γραμμικό Πρόγραμμα είναι επιλύσιμο και έχει βέλτιστη λύση τη $(x_1, x_2, x_3, x_4) = (1, 0 , 1, 0)$ με κόστος -1.

3η Άσκηση

Δίνεται το (μη γραμμικό) πρόγραμμα:

$R = min\ \{\frac{c_1^T x + d_1}{c_2^T x + d_2} : Ax \leq b,\ c_2^T x + d_2 > 0\}$

Έστω ότι η περιοχή των εφικτών λύσεων είναι φραγμένη και ότι η αντικειμενική τιμή της βέλτιστης λύσης ανήκει στο διάστημα [L, U].

1. *Έστω ότι είναι γνωστό ένα δ > 0, τ.ω. για κάθε εφικτή λύση x, $c_2^T x + d_2 \geq \delta$. Να δειχθεί ότι για κάθε ε > 0, μπορεί να υπολογιστεί μια (1 + ε)-προσέγγιση της βέλτιστης λύσης του R. Ποια είναι η υπολογιστική πολυπλοκότητα του αλγόριθμου;*

- Έστω ότι είναι γνωστό ένα $\delta > 0$, τ.ω. Ν εφικτή λύση να ισχύει $c_2^T x + d_2 \geq \delta$. Συνεπώς, αρκούν οι γραμμικοί περιορισμοί $Ax \leq b$, $c_2^T x + d_2 \geq \delta$ για να εκφράσουν όλες τις δυνατές λύσεις.
- Έστω επίσης $K \in [L, U]$. Τότε, $\frac{c_1^T x + d_1}{c_2^T x + d_2} \leq K \Rightarrow c_1^T x + d_1 \leq K \cdot (c_2^T x + d_2)$

Παρατηρείται ότι εάν R > K, τότε δεν υπάρχει εφικτή λύση που να ικανοποιεί και τους 3 παραπάνω περιορισμούς. Εάν όμως, R ≤ K, τότε υπάρχει τουλάχιστον μια λύση που ικανοποιεί και τους 3 παραπάνω περιορισμούς.

Επομένως, το γραμμικό πρόβλημα $\Pi_0(K)$ με περιορισμούς:
$Ax \leq b$, $c_2^T x + d_2 \geq \delta$, $c_1^T x + d_1 \leq K \cdot (c_2^T x + d_2)$
θα είναι ικανοποιήσιμο αν και μόνο αν, R ≤ K.
Μια προσέγγιση του R μπορεί να προκύψει με binary search στο K. Ο σχετικός ψευδοκώδικας είναι ο εξής:

```
// Algorithm (1+ϵ)-approximation of R
function approximate(ϵ)
    start ← L
    end ← U
    while end − start ≥ ϵ do
        mid ← (start + end)/2

        if Π0(mid) is feasible then
            end ← mid
        else
            start ← mid
        end if
    end while

    if Π0(mid) is feasible then
        return mid
    else
        return end
    end if

end function
```

Σε κάθε επανάληψη του αλγόριθμου ισχύει ότι OPT∈[start, end] και τερματίζει ότι end – start < ε. Επομένως, επειδή end ≥ SOL ≥ OPT ≥ start, ισχύει ότι SOL – OPT ≤ ε ⇒ SOL ≤ OPT + ε .

<u>*Πολυπλοκότητα*</u>: Η δυαδική αναζήτηση εκτελεί $\log(\frac{U-L}{\varepsilon})$ επαναλήψεις και σε κάθε επανάληψη λύνει ένα γραμμικό πρόβλημα (όπου κάθε τέτοιο πρόβλημα, λύνεται σε πολυωνυμικό χρόνο). Συνεπώς, με δοσμένα τα L, U και δ, το πρόβλημα αυτό ανήκει στην FPTAS.

2. *Να λυθεί το R όπως στο (1), αλλά χωρίς την υπόθεση σχετικά με την ύπαρξη του δ. Τώρα είναι γνωστό μόνο ότι:* $c_2^T x + d_2 > 0$.

Εφόσον δεν είναι γνωστή η υπόθεση σχετικά με την ύπαρξη του δ, ώστε να ισχύει ότι $c_2^T x + d_2 \geq$ δ για κάθε δυνατή λύση, τότε το Π_0 δεν εκφράζει όλες τις εφικτές λύσεις.
Το μόνο γνωστό είναι ότι : $c_2^T x + d_2 > 0$ και σύμφωνα με αυτό αναζητείται η ύπαρξη εφικτής λύσης του Π_0.
Θέτοντας ως objective function την : $c_2^T x$, προκύπτει ένα νέο γραμμικό πρόβλημα $\Pi_1(K)$, το οποίο θα είναι:

maximize: $c_2^T x$
subject to: $Ax \leq b$
$c_1^T x + d_1 \leq K \cdot (c_2^T x + d_2)$

Τότε, εάν το $\Pi_1(K)$ είναι infeasible, θα ισχύει σίγουρα ότι K < R.

Επίσης, εάν έχει βέλτιστη λύση με τιμή A, τότε K ≥ R αν και μόνο αν $A + d_2 > 0$. Αυτό ισχύει διότι, αν $A + d_2 \leq 0$, τότε ∀ εφικτή λύση του $\Pi_1(K)$, το $c_2^T x + d_2$ είναι αρνητικό οπότε αυτές οι λύσεις δεν θα αποτελούν εφικτές λύσεις του αρχικού προβλήματος. Αν όμως, $A + d_2 > 0$, τότε υπάρχουν εφικτές λύσεις του $\Pi_1(K)$ που θα αποτελούν και εφικτές λύσεις για το αρχικό πρόβλημα.

Συνεπώς, η λύση θα είναι ξανά ένας binary search αλγόριθμος με μόνη διαφορά ότι τώρα θα πραγματοποιηθεί με το Π_1.

4η Άσκηση

*Έστω Α ένας πίνακα m × n, **x** ένα n-διάνυσμα μεταβλητών, και **b** ένα m-διάνυσμα. Το γραμμικό σύστημα Ax ≤ b καλείται μη-συμβιβαστό αν υπάρχει m-διάνυσμα y τέτοιο ώστε $A^Ty = 0$, $b^Ty < 0$, και $y \geq 0$. Ζητείται να αποδειχθεί ότι το σύστημα Ax ≤ b είναι μη-επιλύσιμο αν και μόνο αν είναι μη-συμβιβαστό.*

<u>*Ορισμός*</u>: Ένα σύστημα Ax ≤ b καλείται μη-συμβιβαστό αν υπάρχει m-διάνυσμα y τ.ω. $A^Ty=0$, $b^Ty<0$.

- <u>*Αντίστροφο*</u>

<u>*Πρόταση*</u>: Αν ∃ y ≥ 0: $A^Ty = 0$ και $b^Ty < 0$, τότε Ax ≤ b είναι μη-επιλύσιμο.

<u>*Απόδειξη*</u>: Έστω ότι το Ax ≤ b είναι επιλύσιμο, δηλαδή υπάρχει κάποιο x' τέτοιο ώστε Ax' ≤ b. Τότε ισχύει: $Ax' \leq b \Rightarrow x'^TA^T \leq b^T \Rightarrow x'^TA^Ty \leq b^Ty \xrightarrow{A^Ty=0} b^Ty \geq 0$

Όμως, από υπόθεση $b^Ty < 0$. Οπότε προκύπτει άτοπο. Άρα, το Ax ≤ b είναι μη-επιλύσιμο.

- <u>*Ευθύ*</u>

<u>*Πρόταση*</u>: Αν Ax ≤ b είναι μη-επιλύσιμο, τότε ∃ y ≥ 0 τ.ω. $A^Ty = 0$ και $b^Ty < 0$.

<u>*Απόδειξη*</u>: Θα γίνει χρήση του Projection Theorem, σύμφωνα με το οποίο "Έστω K ένα κλειστό, κυρτό και μη-κενό σύνολο του $\mathbb{R}^n$ και b κάποιο σημείο του $\mathbb{R}^n$. Η προβολή του b στο K είναι ένα σημείο p του K που έχει την ελάχιστη απόσταση από το b (||b-p||). Δηλαδή, για κάθε zϵK, $(z-p)^T(b-p) \leq 0$".

Έστω το σύνολο $K = \{d\epsilon\mathbb{R}^m: \exists x\, \epsilon\mathbb{R}^n$ τ.ω. $Ax \leq d\}$. Εφόσον το Ax ≤ b είναι μη-επιλύσιμο, τότε b∉K. Έστω επίσης, p η προβολή του b στο K. Επειδή pϵK, από τον ορισμό, υπάρχει κάποιο $w\epsilon\mathbb{R}^n$ για το οποίο ισχύει Aw ≤ b.

Οπότε, από το Projection Theorem, ∀ zϵK θα ισχύει $(z-p)^T(b-p) \leq 0$.

Συνεπώς, για κάθε x ισχύει ότι: $(z-p)^T(b-p) \leq 0$ και $Ax \leq z \Rightarrow (Ax-p)^T(b-p) \leq 0$. Θέτοντας, y = b – p, το παραπάνω γράφεται: $(Ax-p)^Ty \leq 0$.

Επιπλέον, για κάθε x ισχύει ότι: $(Ax-p)^Ty \geq 0$ και $Aw \leq p \Rightarrow (Ax-Aw)^Ty \geq 0 \Rightarrow (x-w)^TA^Ty \geq 0$. (*)

Ακόμη, ως δ_i ορίζεται το διάνυσμα με όλα τα στοιχεία ίσα με 0 εκτός από το i-οστό που είναι ίσο με 1: $\delta_i = \begin{pmatrix} 0 \\ \dots \\ 0 \\ 1 \\ 0 \\ \dots \\ 0 \end{pmatrix}$

Θέτοντας $x = w + \delta_i$ στην σχέση (*), προκύπτει για την i-οστή συντεταγμένη του A^Ty ότι είναι μη αρνητική, δηλ. $(A^Ty)_i \geq 0$. Αντίστοιχα, θέτοντας $x = w - \delta_i$, προκύπτει ότι $(A^Ty)_i \leq 0$.

Επομένως, $(A^Ty)_i = 0$ για κάθε iϵ[n], δηλ. $A^Ty = 0$.

Εκκρεμεί να δειχθεί ότι για το συγκεκριμένο y ισχύει ότι $b^Ty < 0$:

$b^Ty = (p-y)^Ty = p^Ty - y^Ty$ και $(Ax-p)^Ty \geq 0, \forall x \xrightarrow{x=0} p^Ty \leq 0 \xrightarrow{y^Ty \geq 0} b^Ty < 0$

■

5η Άσκηση

Έστω ο παρακάτω αλγόριθμος προσέγγισης για το πρόβλημα Vertex Cover σε ένα γράφημα G(V, E), όπου κάθε κορυφή v∈V έχει βάρος w(v) > 0. Το ζητούμενο είναι ο υπολογισμός ενός συνόλου κορυφών C ⊆V, με ελάχιστο συνολικό βάρος, ώστε κάθε ακμή e∈E να έχει τουλάχιστον ένα άκρο της στο C.

```
WeightedVertexCover(G(V,E), w)
    C ← ∅;
    ∀v ∈ V , t(v) ← w(v);
    ∀e ∈ E, c(e) ← 0;
    while C δεν είναι vertex cover do
        e = {u, v} μια ακάλυπτη ακμή;
        δ ← min{t(u), t(v)};
        t(u) ← t(u) – δ;
        t(v) ← t(v) – δ;
        c(e) ← δ;
        C ← C ∪ {z ∈ {u, v} | t(z) = 0};
    end while
return(C);
```

Ζητείται να δειχθεί ότι ο αλγόριθμος επιτυγχάνει λόγο προσέγγισης 2. Ειδικότερα, να δειχθεί ότι για τα C και c(e), e∈E, στο τέλος του αλγορίθμου, ισχύουν τα εξής:

1. *Το C καλύπτει όλες τις ακμές.*

   Εφόσον ο αλγόριθμος τερματίζει όταν το C γίνει Vertex Cover (επειδή για κάθε ακάλυπτη ακμή προσθέτει το ένα άκρο της στο C), το C καλύπτει όλες τις ακμές.

2. *Το συνολικό βάρος του C είναι μικρότερο ή ίσο του* $2\sum_{e\in E} c(e)$.

   Σε κάθε επανάληψη του βρόχου ισχύει ότι ∀ v∈V: $\sum_{u\in e:\, e\in E} c(e) = w(u) - t(u)$
   Αυτό αποδεικνύεται επαγωγικά.
   Πριν την πρώτη επανάληψη ισχύει t(u) = w(u), ∀ u∈V και c(e) = 0, ∀ e∈E. Σε κάθε επανάληψη, το t(u) μειώνεται κατά δ και μία προσπίπτουσα ακμή από 0 γίνεται δ, επομένως η ιδιότητα διατηρείται.
   Για κάθε κορυφή του C ισχύει t(u) = 0, οπότε: ∀ v∈C: $\sum_{u\in e:\, e\in E} c(e) = w(u)$
   Επομένως, για το συνολικό βάρος του C ισχύει:
   $\sum_{u\in C} w(u) = \sum_{u\in C} \sum_{u\in e:\, e\in E} c(e) \le \sum_{v\in V} \sum_{u\in e:\, e\in E} c(e) = 2\sum_{e\in E} c(e)$

3. *Το συνολικό βάρος της βέλτιστης λύσης είναι μεγαλύτερο ή ίσο του* $\sum_{e\in E} c(e)$. *(hint: LP duality)*

   Αρκεί να δειχθεί ότι το συνολικό βάρος της βέλτιστης λύσης έχει ως κάτω φράγμα το $\sum_{e\in E} c(e)$.
   Το weighted vertex cover λύνεται βέλτιστα από το παρακάτω Integer Program:
   minimize: $w^T x$
   subject to: $x_u + x_v \ge 1$ ∀ (u, v)∈E
   $x_i \in \{0, 1\}$

Το πρόγραμμα αυτό έχει μεγαλύτερη ή ίση λύση από το relaxed LP:
minimize: $w^T x$
subject to: $x_u + x_v \geq 1 \; \forall \; (u, v) \epsilon E$

Με τη βοήθεια του hint, κατασκευάζεται το δυικό πρόβλημα του παραπάνω. Ο πίνακας περιορισμών έχει μια γραμμή για κάθε ακμή και μια στήλη για κάθε κορυφή. Το dual linear program θα έχει μια μεταβλητή για κάθε περιορισμό, δηλαδή μια μεταβλητή για κάθε ακμή. Στην στήλη μιας κορυφής, όλα τα στοιχεία είναι μηδενικά, εκτός από τις γραμμές που αντιστοιχούν σε προσπίπτουσες στην κορυφή ακμές, τα οποία έχουν τιμή 1. Συνεπώς, το εσωτερικό γινόμενο μιας στήλης του A με τις μεταβλητές του δυικού προγράμματος, είναι ουσιαστικά το άθροισμα των μεταβλητών που αντιστοιχούν στις προσπίπτουσες ακμές στην αντίστοιχη κορυφή:
minimize: $\mathbb{1} \cdot y$
subject to: $\sum_{\substack{e \epsilon E: \\ u \epsilon E}} y_e \leq w(u) \; \forall \; u \epsilon V$

Παρατηρείται ότι η λύση $y_e = c(e)$ είναι μια εφικτή λύση του δυικού προβλήματος, αφού:
$\forall \; u \epsilon V : \sum_{\substack{e \epsilon E: \\ u \epsilon e}} c(e) = w(u) - t(u) \xrightarrow{t(u) \geq 0} \forall \; u \epsilon V : \sum_{\substack{e \epsilon E: \\ u \epsilon e}} c(e) \leq w(u)$

Ζητείται ακόμη ένα (κατά το δυνατόν απλούστερο) γράφημα όπου ο αλγόριθμος υπολογίζει ένα Vertex Cover με συνολικό βάρος διπλάσιο από αυτό της βέλτιστης λύσης.

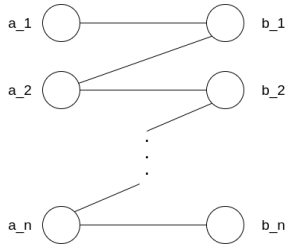
Έστω x* μια λύση του primal LP και w* το βάρος του βέλτιστου vertex cover (δηλαδή η λύση του Integer Program). Επειδή η βέλτιστη λύση του Primal δεν είναι χειρότερη από αυτή του Integer Program και επειδή κάθε λύση του δυικού είναι μικρότερη ή ίση από κάθε λύση του πρωτεύοντος:

$w^* \geq x^* \geq \sum_{e \epsilon E} c(e)$

Από τα ερωτήματα (2) και (3), προκύπτει ότι: $w(C) \leq 2 \cdot w^*$.

Tight Example

Θα δείξουμε ότι το 2 είναι tight ratio για τον παραπάνω αλγόριθμο. Έστω ο παρακάτω γράφος:



Έστω ότι όλες οι κορυφές έχουν βάρος 1, $\forall \; v \epsilon V : w(v) = 1$.

Αρχίζοντας από την κορυφή $\{a_1, b_1\}$, ο αλγόριθμος θα προσθέσει την a1 στο C και θέτει $t(b_1) = 0$. Έπειτα, εξετάζοντας την ακμή $\{a_2, b_1\}$ θα επιλέξει την κορυφή b_1 και το $t(a_2)$ θα μείνει αναλλοίωτο κ.ο.κ. έως την κορυφή an που θα είναι η τελευταία που θα προστεθεί στο C.

$C = \{a_1, \ldots, a_n, b_1, \ldots, b_{n-1}\}$, $SOL = 2n - 1$ & $C^* = \{a_1, \ldots, a_n\}$, $OPT = n$

$$\frac{SOL}{OPT} = 2 - \frac{1}{n}, \; \forall \; n \epsilon \mathbb{N}$$

6η Άσκηση

(a) Ζητείται ο σχεδιασμός αλγόριθμου που να λύνει βέλτιστα το πρόβλημα Vertex Cover όταν ο γράφος εισόδου είναι δένδρο.

Όταν ο γράφος εισόδου, για το πρόβλημα Vertex Cover, είναι δέντρο, τότε το πρόβλημα λύνεται βέλτιστα σε πολυωνυμικό χρόνο. Για την επίλυση θα γίνει χρήση δυναμικού προγραμματισμού. Αρχικά, επιλέγεται ένας αυθαίρετος κόμβος του δέντρου και αυτός θα θεωρείται η ρίζα. Έπειτα ορίζεται το state: dp[u][b], uϵV, bϵ{0, 1} ως το min vertex cover του υποδέντρου που έχει ως ρίζα τον u και με το u να ανήκει αναγκαστικά στο cover, αν και μόνο αν b = 1. Αν η ρίζα δεν ανήκει στο vertex cover, τότε όλα τα παιδιά της θα ανήκουν αναγκαστικά στο cover. Αντίθετα, αν η ρίζα του υποδέντρου ανήκει στο vertex cover, τότε τα παιδιά της δεν χρειάζεται να ανήκουν στο cover αφού οι ακμές προς αυτά καλύπτονται.

```
// Memoization table initialized to -1
// memo[|V|][2]

function DP(u, b)
    if memo[u][b] ≠ -1 then
        return memo[u][n]
    end if
    answer ← b
    for v ∈ adj[u] do
        if b == 1 then
            answer ← answer + min(dp(u, 0), dp(u, 1))
        else
            answer ← answer + dp(u, 1)
        end if
    end for
    return answer
end function

function VERTEXCOVER
    return min(dp(0, 0), dp(0, 1))
end function
```

Πολυπλοκότητα: υπάρχουν συνολικά 2·|V| states και σε κάθε ένα από αυτά γίνεται O(deg(u)) μη-αναδρομική δουλειά. Συνεπώς, η συνολική πολυπλοκότητα είναι:
$\sum_{b\epsilon\{0,1\}} \sum_{u\epsilon V} \deg(u) = 2\cdot|E| = O(|V|)$, αφού $|V| = |E| - 1$

(b) Ζητείται να αποδειχθεί ότι ο παρακάτω αλγόριθμος είναι 2-προσεγγιστικός για το πρόβλημα Vertex Cover στη γενική περίπτωση (είσοδος οποιοσδήποτε γράφος G): Βρες ένα δέντρο T με αναζήτηση κατά βάθος (DFS) στον δοσμένο γράφο G και επίστρεψε το σύνολο των κόμβων S που δεν είναι φύλλα του T.

Ξεκινώντας από έναν οποιοδήποτε κόμβο ενός γράφου G, μπορεί να βρεθεί ένα δέντρο T με DFS.

Αποδεικνύεται αρχικά πως ο αλγόριθμος της εκφώνησης δίνει λύση στο Vertex Cover: Έστω ότι δε δίνει λύση. Τότε θα υπάρχει ακμή e(u, v) που δε θα καλύπτεται, συνεπώς κανένας από τους κόμβους u, v δε θα είναι επιλεγμένος, θα είναι δηλαδή φύλλα. Αυτό είναι άτοπο, καθώς δε μπορούν 2 φύλλα να συνδέονται μεταξύ τους.

Θα αποδειχθεί ότι το σύνολο S των εσωτερικών κόμβων του T είναι 2-προσέγγιση του Vertex Cover του G. Αυτό αποδεικνύεται, χρησιμοποιώντας την έννοια του maximal matching: Είναι γνωστό πως το σύνολο κόμβων της OPT λύσης δε μπορεί να είναι μικρότερης πληθικότητας από το σύνολο των ανεξάρτητων ακμών στο maximal matching (δε θα έφταναν οι κόμβοι για να καλύψουν όλες τις ανεξάρτητες ακμές – χρειάζεται 1 ανά τέτοια ακμή). Έτσι, OPT = $|C^*| \geq |MM|$.

Παράγεται maximal matching στο DFS Tree με την εξής διαδικασία: Αρχίζοντας από τη ρίζα του δέντρου και επιλέγεται μία από τις προσπίπτουσες ακμές και διαγράφεται. Επαναλαμβάνεται η ίδια διαδικασία στα δέντρα που προκύπτουν μέχρι να εξαλειφθούν όλα ή να καταλήξουν σε μοναδικό κόμβο (φύλλο). Έτσι επιτυγχάνεται η κάλυψη κάθε εσωτερικού κόμβου του γραφήματος ενώ τα όποια φύλλα μένουν ακάλυπτα δεν μπορούν έτσι κι αλλιώς να συνδέονται. Συνεπώς δε χάνεται κάποια ακμή και το matching είναι maximal. Στην καλύτερη περίπτωση, κάθε ακμή που επιλέχθηκε, θα καλύπτει 2 κόμβους ενώ θα έχουν μείνει ακάλυπτα όλα τα φύλλα, που είναι αχρείαστα. Συνεπώς $|MM| \geq (|V| - \text{φύλλα})/2 = SOL/2$.

<u>*Εναλλακτικά*</u>, ισχύει ότι το σύνολο των εσωτερικών κόμβων είναι cover του G, αφού καλύπτει όλες τις ακμές μεταξύ των εσωτερικών κόμβων και οι ακμές των φύλλων καλύπτονται από το τελευταίο επίπεδο.

Με *επαγωγή* θα αποδειχθεί ότι το σύνολο S περιέχει matching τουλάχιστον $|S|/2$ ακμών.
Αν το T έχει μόνο έναν κόμβο, τότε η ιδιότητα ισχύει. Έστω ότι έχει n κόμβους και κάθε δέντρο με το πολύ n – 1 έχει matching όσο το μισό του μέγεθος. Έστω $T_1, \ldots, T_n$ τα υποδέντρα της ρίζας r του T και $S_1, \ldots, S_m$ τα υποδέντρα της ρίζας r_1 του δέντρου T_1. Χρησιμοποιώντας τα matching των υποδέντρων $T_1, \ldots, T_n, S_1, \ldots, S_m$ και την ακμή (r, r_1) δημιουργείται ένα matching μέγεθος:
$1 + \sum_{i=1}^{m} M(S_i) + \sum_{i=2}^{n} M(T_i)$

Από την επαγωγική υπόθεση:
$1 + \sum_{i=1}^{m} M(S_i) + \sum_{i=2}^{n} M(T_i) \geq 1 + \frac{1}{2}\sum_{i=1}^{m} I(S_i) + \frac{1}{2}\sum_{i=2}^{n} I(T_i) = 1 + \frac{1}{2}(I(T) - 2) = \frac{I(T)}{2}$

Όπου I(B) ο αριθμός των εσωτερικών κόμβων του δέντρου B.

Προφανώς, το Optimal Vertex Cover θα χρειαστεί έναν κόμβο για κάθε ακμή του matching, επομένως: $OPT \geq \frac{|S|}{2} \Rightarrow SOL \leq 2 \cdot OPT$

7η Άσκηση

Ζητείται να αποδειχθεί ότι ο λόγος προσέγγισης που επιτυγχάνει ο Greedy αλγόριθμος για το Weighted Set Cover είναι στην πραγματικότητα $H_{|S_{max}|}$, όπου S_{max} το σύνολο εισόδου με τη μεγαλύτερη πληθικότητα.

Hint: Να αποδειχθεί ότι τα στοιχεία οποιουδήποτε συνόλου SϵS καλύπτονται από τον Greedy με συνολικό κόστος το πολύ $H_{|S|} \cdot cost(S)$, δηλαδή $\sum_{e \epsilon S} price(e) \leq H_{|S|} \cdot cost(S)$.

Ο greedy αλγόριθμος επιλέγει σε κάθε βήμα το σύνολο με τον καλύτερο λόγο κόστους προς στοιχεία που δεν έχουν καλυφθεί. Για την περαιτέρω ανάλυση, τίθεται ως q_e ο λόγος του συνόλου που κάλυψε πρώτο το στοιχείο e.

Λήμμα 1

Έστω ότι ο greedy έχει ήδη καλύψει l αντικείμενα από ένα σύνολο $S_i \epsilon S$. Επειδή διαλέγει το σύνολο με το μικρότερο ratio, το επόμενο ratio που θα διαλέξει θα είναι το πολύ όσο και το ratio του S_i, δηλαδή $\frac{c_i}{|S_i|-l}$.

Λήμμα 2

Το κόστος του greedy είναι ακριβώς $\sum_{e \epsilon u} q_e$. Αυτό ισχύει και ως invariant κατά την εκτέλεση του αλγορίθμου. Στην αρχή όλα τα q_e είναι 0. Όταν επιλέγεται ένα set I που έχει l ακάλυπτα στοιχεία, το κόστος του greedy αυξάνεται κατά c_i και τα q_e των l στοιχείων που δεν έχουν καλυφθεί γίνονται c_i/l. Συνεπώς, το άθροισμά τους αυξάνεται και αυτό κατά c_i.

Από το Λήμμα 1, προκύπτει ότι το j-οστό στοιχείο που καλύπτεται από ένα σύνολο S_i θα έχει q_{e_j} το πολύ $\frac{c_i}{|S_i|-l}$. Οπότε, για κάθε υποσύνολο S_i ισχύει:

$$\sum_{e \epsilon S_i} q_e \leq \frac{c_i}{|S_i|} + \frac{c_i}{|S_i|-1} + \ldots + \frac{c_i}{2} + \frac{c_i}{1} = c_i \cdot H_{|S_i|}$$

Επιπροσθέτως, επειδή η optimal λύση καλύπτει όλα τα στοιχεία τουλάχιστον μια φορά:

$$\sum_{e \epsilon S_i} q_e \leq \sum_{S_i \epsilon OPT} \sum_{e \epsilon S_i} q_e \leq \sum_{S_i \epsilon OPT} c_i \cdot H_{|S_i|} \leq H_{|S_{max}|} \sum_{S_i \epsilon OPT} c_i = H_{|S_{max}|} \cdot OPT$$

$$SOL \leq H_{|S_{max}|} \cdot OPT$$

8η Άσκηση

(a) Ζητείται ο σχεδιασμός απλού f-προσεγγιστικού αλγορίθμου για το πρόβλημα Cardinality Set Cover, όπου f είναι το μέγιστο πλήθος συνόλων στα οποία μπορεί να ανήκει κάποιο στοιχείο. Επιπλέον, ζητείται η ορθότητά του και ο λόγος προσέγγισης που επιτυγχάνει. Τέλος, ζητείται η εύρεση tight example για τον λόγο προσέγγισης του αλγορίθμου.
Hint: παρατηρήστε ποιο είναι το f στο πρόβλημα (Cardinality) Vertex Cover και γενικεύστε κατάλληλα.

Έστω ένα σύνολο U = {u_1, …, u_n} και μια συλλογή υποσυνόλων του U, S = {S_1, …, S_m}.
Αναζητείται ο ελάχιστος αριθμός συνόλων που καλύπτουν όλο το U.
Ως f ορίζεται ο μέγιστος αριθμός συνόλων στα οποία ανήκει κάποιο στοιχείο του U:
$f = \max_{u \in U} |\{i : u \in S_i\}|$
Ως matching στοιχείων του U ορίζεται ένα σύνολο στοιχείων τα οποία ανά δύο ανήκουν σε διαφορετικά υποσύνολα. Maximal matching ονομάζεται ένα matching στο οποίο δεν μπορεί να προστεθεί κανένα άλλο στοιχείο.
Διαλέγοντας greedily στοιχεία, βρίσκεται ένα maximal matching M και επιστρέφονται όλα τα υποσύνολα στα οποία ανήκουν τα στοιχεία του M:
SOL = {$S_i \in S : \exists u \in M$ s.t. $u \in S_i$}

Επεξήγηση
Το SOL είναι Set Cover αφού αν δεν ήταν, θα υπήρχε κάποιο στοιχείο του U που δεν θα ανήκε σε κανένα σύνολο της λύσης, άρα θα μπορούσε να προστεθεί και στο matching M, κάτι που είναι άτοπο αφού το M ήταν maximal matching.
Για κάθε maximal matching M ισχύει: OPT ≥ |M|, αφού τα στοιχεία του M ανήκουν σε ξεχωριστά σύνολα και απαιτείται τουλάχιστον ένα σύνολο για να καλυφθεί κάθε στοιχείο του M.
Ακόμη, κάθε στοιχείο ανήκει το πολύ σε f σύνολα, οπότε SOL ≤ f · |M| .
Συνεπώς, προκύπτει ότι SOL ≤ f · OPT .

Tight Example
Το vertex cover είναι ειδική περίπτωση του set cover, με τις ακμές να είναι τα στοιχεία του U και τους κόμβους να είναι σύνολα των προσπιπτουσών ακμών. Επομένως, ένα tight example αποτελούν το vertex cover τους πλήρεις διμερείς γράφους $K_{n,n}$. Εκεί το maximal matching θα είναι n ακμές και δίνονται ως απάντηση όλοι οι κόμβοι 2n ενώ αρκούν οι n για να καλυφθούν όλες οι ακμές.

(b) Ζητείται ο σχεδιασμός f-προσεγγιστικού αλγορίθμου για το πρόβλημα Weighted Set Cover. Επιπλέον, ζητείται η ορθότητά του και ο λόγος προσέγγισης που επιτυγχάνει. Τέλος, ζητείται η εύρεση tight example για τον λόγο προσέγγισης του αλγορίθμου.
Hint: γενικεύστε την ιδέα του degree-weighted αλγορίθμου για το Weighted Vertex Cover. Αν θέλετε, μπορείτε να αποδείξετε τον λόγο προσέγγισης προσαρμόζοντας την τεχνική της 5ης άσκησης.

Ως προέκταση του Cardinality Set Cover, έστω ακόμη μια συνάρτηση w: S → ℝ που δίνει το βάρος κάθε διαθέσιμου υποσυνόλου. Αναζητείται η εύρεση του Set Cover με το ελάχιστο συνολικό βάρος.

Αρχικά, επιλέγονται άπληστα τα σύνολα που έχουν το ελάχιστο κόστος ανά καλυπτόμενο στοιχείο. Συγκεκριμένα, στο στάδιο 0, τίθεται $B_0 = S$, όπου S η αρχική συλλογή υποσυνόλων. Έπειτα, αφαιρούνται τα κενά σύνολα και ονομάζεται η συλλογή των μη-κενών D_0. Στη συνέχεια, βρίσκεται το ελάχιστο κόστος ανά στοιχείο: $c = \min_{S_i \in D_0} w(S_i)/|S_i|$. Ύστερα, ανανεώνονται τα βάρη των συνόλων ως εξής: $w'(S_i) = w(S_i) - t_0(S_i)$, όπου $t_0(S_i) = c \cdot |S_i|$. Μετά, επιλέγονται τα σύνολα που έχουν πλέον μηδενικό βάρος, το σύνολο των οποίων ονομάζεται W_0. Τέλος, προστίθεται το W_0 στο cover και συνεχίζεται η διαδικασία από την αρχή αφαιρώντας από όλα τα σύνολα τα στοιχεία που καλύφθηκαν ώσπου όλα τα σύνολα μείνουν κενά.

Επεξήγηση

Η λύση θα είναι set cover, αφού ο αλγόριθμος τερματίζει όταν όλα τα διαθέσιμα σύνολα είναι άδεια και σε κάθε βήμα αφαιρούνται μόνο τα στοιχεία που καλύπτονται.

Θα ονομαζόταν η συνάρτηση βάρους $w: S \rightarrow \mathbb{R}$ cardinality-weighted αν και μόνο αν για κάθε $S_i \in S$ υπάρχει μια σταθερά c ώστε $w(S_i) = c \cdot |S_i|$.

Θα δειχθεί ότι για κάθε cardinality-weighted συνάρτηση, ισχύει ότι $w(S) \leq f \cdot w(K)$, για κάθε set cover K.
Έστω K ένα weighted set cover.
Επειδή το K καλύπτει όλα τα στοιχεία του U, $\sum_{s \in K} |s| \geq |U| \Rightarrow w(K) \geq c \cdot w(K)$.
Ακόμα, επειδή κάθε στοιχείο ανήκει το πολύ σε f σύνολα, $\sum_{s \in S} |s| \leq f \cdot |U| \Rightarrow w(S) \leq fc \cdot |U|$.
Από τις δύο προηγούμενες σχέσεις, $w(S) \leq f \cdot w(K)$.

Με άλλα λόγια, σε κάθε βήμα επιλέγεται ένα υποσύνολο των συνόλων για τα οποία η συνάρτηση βάρους είναι τοπικά cardinality-weighted με μικρό c. Πιο διαισθητικά, εφαρμόζεται decompose της συλλογής συνόλων σε cardinality-weighted συλλογές όπου λόγω της προηγούμενης απόδειξης, το να επιλεγούν όλα τα σύνολα θα είναι κοντά στη βέλτιστη λύση. Εφόσον, για κάθε μικρή συλλογή, πλησιάζει κοντά στην βέλτιστη λύση της συλλογής, τότε και συνολικά θα πλησιάζει κοντά στην βέλτιστη λύση του προβλήματος.

Έστω S* ένα optimal set cover και έστω $s \in SOL$ ένα σύνολο που διάλεξε η δική μας λύση. Τότε, $s \in W_j$ για κάποιο j: $w(s) = \sum_{i \leq j} t_i(s)$
Αν το σύνολο s δεν ανήκει στη λύση μας, τότε θα ανήκει σε κάποιο D_j. Οπότε, το βάρος του είναι κάτω φραγμένο από: $w(s) \geq \sum_{i \leq j} t_i(s)$
Για κάθε υποσυλλογή συνόλων που δημιουργείται, επειδή είναι cardinality-weighted, θα ισχύει η σχέση που αποδείχθηκε παραπάνω.
Επιπλέον, το $S^* \cap W_j$ είναι set cover για τα στοιχεία που καλύφθηκαν στο j-οστό επίπεδο.
Επομένως, με βάση τις σχέσεις για τα βάρη των συνόλων και το λήμμα που αποδείχθηκε:
$w(SOL) = \sum_{i=0}^{k} t_i(SOL \cap D_i) \leq f \sum_{i=0}^{k} t_i(S^* \cap D_i) \leq f \cdot w(S^*)$

Tight Example

Με αντίστοιχο τρόπο με το προηγούμενο ερώτημα, tight example αποτελούν οι πλήρεις διμερείς γράφοι $K_{n,n}$ με unit weights. Στην πρώτη επανάληψη του αλγορίθμου, θα επιλεχθούν όλοι οι κόμβοι (η συνάρτηση βάρους είναι degree-weighted για $c = 1/n$) ενώ η βέλτιστη απάντηση είναι οι n κόμβοι.

9η Άσκηση

(a) Ζητείται η συμπλήρωση της απόδειξης που βρίσκεται στις διαφάνειες για τον λόγο προσέγγισης 5/3 για το πρόβλημα Metric $TSP_{(s,t)\text{-}path}$. Επίσης, ζητείται η εξήγηση του ρόλου του όρου $c_{s,t}$ στην ανάλυση καθενός από τους δύο επιμέρους αλγορίθμους.

Αλγόριθμος

i) Βρίσκει ελάχιστο συνδετικό δέντρο T στον γράφο G. Έπειτα, διπλασιάζει τις ακμές του T και αφαιρεί ένα (s, t)-path από το διπλασιασμένο δέντρο. Στη συνέχεια βρίσκει (s, t)-Euler path $p_{s,t}$ και εκτελεί short-cutting για να βρει (s, t)-Hamilton path κόστους: $SOL_1 \leq 2\ OPT_{s,t} - c_{s,t}$.

ii) Βρίσκει ελάχιστο συνδετικό δέντρο T στον γράφο G. Έπειτα, βρίσκει perfect matching M στους κόμβους που πρέπει να αλλάξουν βαθμό ώστε το γράφημα T∪M να έχει (s, t)-Euler path. Στη συνέχεια, με short-cutting βρίσκει (s, t)-Hamilton path κόστους: $SOL_2 \leq \frac{3OPT_{s,t} + c_{s,t}}{2}$.

iii) Επιστρέφει SOL = min (SOL_1, SOL_2).

Επεξήγηση

1) Θα δειχθεί το (i).

Λήμμα 1

Έστω T ένα ελάχιστο συνδετικό δέντρο του G. Ισχύει ότι $OPT_{s\text{-}t} \geq w(T)$.
Το $OPT_{s\text{-}t}$ είναι ένα path που ξεκινά από το s, περνά από όλες τις κορυφές του G και καταλήγει στην t. Δηλαδή είναι ένα συνδετικό δέντρο του G, επομένως εξ' ορισμού το ελάχιστο συνδετικό δέντρο θα έχει μικρότερο ή ίσο βάρος από το $OPT_{s,t}$.

Έστω H το διπλασιασμένο T από το οποίο έχει αφαιρεθεί ένα s-t path. Το H έχει s-t Euler path $p_{s,t}$, αφού διπλασιάζοντας τις ακμές του T όλοι οι κόμβοι έχουν άρτιο βαθμό, και αφαιρώντας ένα s-t path, οι κόμβοι s και t θα έχουν περιττό βαθμό και όλοι οι υπόλοιποι άρτιο (αφού αφαιρούνται 2 ακμές από κάθε ενδιάμεσο κόμβο του s-t path).
Το s-t path που αφαιρέθηκε έχει βάρος μεγαλύτερο ή ίσο της ακμής $c_{s,t}$ λόγω της τριγωνικής ανισότητας. Επομένως, για το H ισχύει: $w(H) \leq 2w(T) - c_{s,t}$
Λόγω του short-cutting, $SOL_1 \leq w(H)$.
Επομένως, $SOL_1 \leq 2w(T) - c_{s,t} \xrightarrow{Λήμμα\ 1} SOL1 \leq 2\ OPT_{s,t} - c_{s,t}$

2) Θα δειχθεί το (ii).

Λήμμα 2

Έστω ένας πλήρης γράφος G(V, E) με άρτιο πλήθος κόμβων και ένα minimum cost perfect matching M των κόμβων. Τότε, ισχύει ότι $w(M) \leq (OPT_{s,t} + c_{s,t})/2$.
Το $OPT_{s,t}$ μαζί με την ακμή $c_{s,t}$ δημιουργούν έναν κύκλο n κόμβων. Οι μισές ακμές του κύκλου είναι ένα matching των κόμβων και έτσι θα είναι βαρύτερο (ή ίσο) από το ελάχιστο matching.

Αρχικά, επιλέγεται ένα ελάχιστο συνδετικό δέντρο T στο γράφημα. Το ζητούμενο είναι να συμπληρωθούν ακμές ώστε όλοι οι κόμβοι να έχουν άρτιο βαθμό εκτός από τις s και t. Έπειτα, επιλέγεται ένα min cost perfect matching των κόμβων που πρέπει να αλλάξει ο βαθμός τους (αυτοί θα είναι σίγουρα άρτιοι στο πλήθος, γεγονός που φαίνεται εύκολα από την διάκριση των 4 περιπτώσεων για τους βαθμούς των s, t και την παρατήρηση ότι ένα γράφημα δεν γίνεται να έχει περιττό αριθμό κορυφών με περιττό βαθμό). Το νέο γράφημα θα έχει $p_{s\text{-}t}$ Euler part, το οποίο με short-cutting γίνεται Hamilton path (SOL_2).
Λόγω του short-cutting, $SOL_2 \leq p_{s_t}$ = w(T) + w(M) .

Από τα Λήμματα 1 & 2, προκύπτει:
$SOL_2 \leq w(T) + w(M) \leq OPT_{s,t} + \frac{OPT_{s,t} + c_{s,t}}{2} \Rightarrow SOL_2 \leq \frac{3OPT_{s,t} + c_{s,t}}{2}$

3) Θα δειχθεί το (iii).

Η επιλογή της λύσης εξαρτάται αποκλειστικά από το $c_{s,t}$:

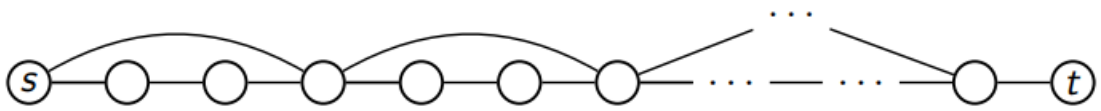
$SOL_1 \leq SOL_2 \Rightarrow 2\ OPT_{s,t} - c_{s,t} \leq \frac{3OPT_{s,t} + c_{s,t}}{2} \Rightarrow \frac{3c_{s,t}}{2} \geq \frac{OPT_{s,t}}{2} \Rightarrow c_{s,t} \geq \frac{OPT_{s,t}}{3}$
Οπότε, διακρίνονται οι εξής περιπτώσεις:

- $c_{s,t} \geq \frac{OPT_{s,t}}{3}$: Δηλ. $SOL = SOL_1$: $SOL = 2\ OPT_{s,t} - c_{s,t} \leq 2\ OPT_{s,t} - \frac{OPT_{s,t}}{3} = \frac{5}{3} OPT_{s,t}$
- $c_{s,t} < \frac{OPT_{s,t}}{3}$: Δηλ. $SOL = SOL_2$: $SOL = \frac{3OPT_{s,t} + c_{s,t}}{2} \leq \frac{3}{2} OPT_{s,t} + \frac{1}{6} OPT_{s,t} = \frac{5}{3} OPT_{s,t}$

Συνεπώς, σε κάθε περίπτωση, $SOL \leq \frac{5}{3} OPT_{s,t}$

(b) Ζητείται να δοθούν tight example για τους επιμέρους αλγορίθμους, καθώς και για τον συνολικό αλγόριθμο.

<u>*Tight Example*</u>: Θα δείχθεί tight example για το οποίο και οι δύο επιμέρους αλγόριθμοι, συνεπώς και ο συνολικός, εμφανίζουν τον ζητούμενο λόγο προσέγγισης 5/3:



Έστω το παραπάνω πλήρες γράφημα με ακμές μοναδιαίου κόστους. Όσες ακμές δε φαίνονται έχουν κόστος ανάλογο του μήκους ενός μονοπατιού από τις ήδη υπάρχουσες. Υποθέτοντας n τριάδες, το βέλτιστο s-t μονοπάτι Euler θα είναι η ευθεία με κόστος $3n$. Ένα ελάχιστο συνδετικό δέντρο για το γράφημα:

Σύμφωνα με τον πρώτο αλγόριθμο θα διπλασιάζονται οι ακμές του γραφήματος και θα αφαιρείται το min s-t path του MST που εν προκειμένω αποτελείται από τις εξωτερικές ακμές. Το Euler path θα έχει την εξής μορφή:

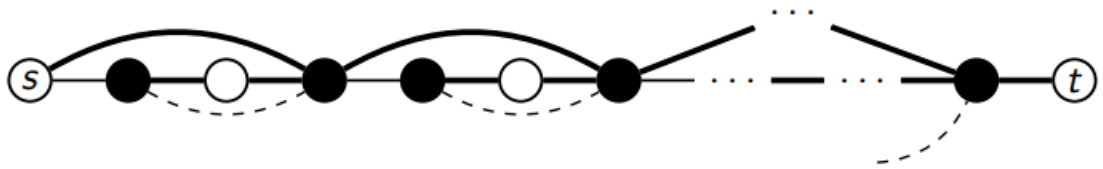
$$1 \equiv s \rightarrow 4 \rightarrow 3 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 7 \rightarrow 6 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow \cdots \rightarrow t$$

Στον κόμβο 2 θα εκτελεστεί short-cutting: θα συνδεθούν 2 και 7, αγνοώντας έτσι τους 3,4. Η ακμή 2-7 θα έχει ωστόσο κόστος 3. Συνολικά το κόστος θα έχει την εξής μορφή:

$$cost = 1+1+1+3+1+1+3+\cdots+3 = \cdots = 5(n-1) + 4 = 5n - 1$$

Συνεπώς προσεγγίζεται ασυμπτωτικά ο ζητούμενος λόγος 5/3.

Σύμφωνα με τον δεύτερο αλγόριθμο, αφού δημιουργηθεί το παραπάνω MST θα πρέπει να σημειωθούν οι κόμβοι περιττού βαθμού (εκτός των s,t) και να βρεθεί ένα perfect matching:



Το κόστος είναι 2 για τις διακεκομμένες ακμές και το συνολικό κόστος είναι $5n$ για το μονοπάτι. Εκτελώντας short-cutting όπως προηγουμένως για την αποφυγή επανάληψης κόμβου, η κατάληξη είναι και πάλι σε κόστος $5n-1$, δηλαδή σε λόγο προσέγγισης 5/3. Συνεπώς ο λόγος είναι 5/3 και για τους 2 επιμέρους αλγόριθμους. Τελικά επιλέγεται $SOL = \min\{SOL_1, SOL_2\} = 5/3\ OPT$, δηλαδή και ο συνολικός αλγόριθμος θα έχει τον ζητούμενο λόγο.

10η Άσκηση

Έστω ο αλγόριθμος για το πρόβλημα Knapsack: τα στοιχεία ταξινομούνται σε φθίνουσα σειρά p(i)/w(i) και εισάγονται με αυτή τη σειρά στο σακίδιο. Όποια στοιχεία δεν χωρούν απορρίπτονται.

a) *Δείξτε ότι ο λόγος προσέγγισης αυτού του αλγορίθμου δεν φράσσεται από καμία σταθερά.*
b) *Δείξτε ότι με την εξής απλή τροποποίηση ο αλγόριθμος γίνεται (1/2)-προσεγγιστικός: επιλέξτε την καλύτερη λύση μεταξύ της παραπάνω διαδικασίας και του να πάρει κανείς μόνο το στοιχείο με τη μεγαλύτερη αξία.*
c) *Γενικεύστε την ιδέα του (β) ώστε να πάρετε ένα PTAS για το πρόβλημα .*

Επεξήγηση

a-b) Για το πρόβλημα Knapsack, ένας απλός greedy algorithm είναι ικανός να υπολογίσει μια καλή προσέγγιση για την βέλτιστη λύση. Το πρόβλημα ορίζεται ως εξής:

Είσοδος:

- n αντικείμενα, όπου κάθε αντικείμενο i∈[n] έχει: cost $c_i \in \mathbb{Z}_+$ και value $v_i \in \mathbb{Z}_+$
- budget $B \in \mathbb{Z}_+$

Ο στόχος είναι η εύρεση ενός υποσυνόλου S αντικειμένων, των οποίων το cost $c(S) = \sum_{i \in S} c_i$ είναι μικρότερο του B και το value $v(S) = \sum_{i \in S} v_i$ είναι maximal.

Παράδειγμα

Έστω το παρακάτω παράδειγμα, με τα αντικείμενα να δίνονται από τον παρακάτω πίνακα.

	α_1	α_2	α_3	α_4
v_i	2	1	7	2
c_i	1	3	2	2

Και το budget B = 4. Έστω επίσης οι ακόλουθες τρεις λύσεις:

Feasible = $\begin{cases} \{a_1, a_2\} \\ v\{a_1, a_2\} = 3 \\ c\{a_1, a_2\} = 4 \end{cases}$, Feasible = $\begin{cases} \{a_1, a_3\} \\ v\{a_1, a_3\} = 9 \\ c\{a_1, a_3\} = 3 \end{cases}$, Not Feasible = $\begin{cases} \{a_1, a_2, a_3\} \\ v\{a_1, a_2, a_3\} = 10 \\ c\{a_1, a_2, a_3\} = 6 \end{cases}$

Η τελευταία λύση δεν είναι feasible επειδή ξεπερνά το budget. Από τις δύο πρώτες λύσεις, η δεύτερη είναι καλύτερη επειδή το value είναι μεγαλύτερο (έχει maximal value μεταξύ των feasible λύσεων).

Μαθηματικό πρόγραμμα για Knapsack

minimize: $v^T x$
subject to: $\sum_{i=1}^{n} c_i x_i \leq B$
$x_i \in \{0, 1\}$

Είναι γνωστό ότι το Knapsack είναι NP-complete optimization πρόβλημα και 'είναι αρκετά αισιόδοξο' να λυθεί σε πολυωνυμικό χρόνο. Για αυτό το λόγο, το πρόβλημα μετατρέπεται στον σχεδιασμό ενός προσεγγιστικού αλγορίθμου.

Ορισμός

Για ένα maximization problem, ο αλγόριθμος λέγεται α-προσεγγιστικός αν για οποιαδήποτε είσοδο, ο αλγόριθμος επιστρέφει feasible solution S τέτοια ώστε: $\frac{f(S)}{f(O)} \geq$ α, όπου O είναι μια βέλτιστη λύση και η f υπολογίζει την ποιότητα αυτή της λύσης.

Fractional View του Knapsack

Είναι γνωστή η Linear Programming relaxation του Knapsack προβλήματος και έχει αποδειχθεί ότι η βέλτιστη λύση της relaxation περιγράφεται ως εξής:

- Γίνεται ταξινόμηση των αντικειμένων κατά φθίνουσα σειρά με βάση το density, όπου ως density ενός στοιχείου i ορίζεται από το λόγο v_i/c_i.
- Έπειτα, προστίθενται τα αντικείμενα στην λύση ένα προς ένα με αυτή τη σειρά εφόσον το άθροισμα των δεν ξεπερνά το budget. Για το πρώτο αντικείμενο i που θα ξεπεράσει το budget, προστίθεται ένα κλάσμα x_i αυτού τέτοιο ώστε το budget constraint να είναι tight. Το κλάσμα x_i ορίζεται ως: $x_i = \frac{B - \sum_{j<i} c_j}{c_i}$

Αλγόριθμος Προσέγγισης για Knapsack

Ο greedy algorithm του original Knapsack είναι:

```
S ← ∅
while c(S ∪ argmax_{i∉S} v_i/c_i) ≤ B do
      S ← S ∪ argmax_{i∉S} v_i/c_i
end while
return S
```

Αυτός ο αλγόριθμος δεν αρκεί για να δώσει μια σταθερή προσέγγιση στην βέλτιστη λύση. Για παράδειγμα, έστω δύο αντικείμενα:

- Το αντικείμενο 1 έχει cost 1 και value 2
- Το αντικείμενο 2 έχει cost M και value M (M > 2)
- Το budget είναι B

Ο greedy algorithm θα επιλέξει το 1 (έχει καλύτερο density) και θα επιστρέψει αφού το αντικείμενο 2 δεν μπορεί να προστεθεί χωρίς να ξεπεράσει το budget. Όμως, είναι ξεκάθαρο ότι η βέλτιστη λύση είναι να επιλέξει το αντικείμενο 2. Ο λόγος ανάμεσα στο value της λύσης που επέστρεψε ο greedy algorithm και την βέλτιστη λύση είναι 1/M το οποίο μικραίνει καθ0ώς το M τείνει στο άπειρο.

Μια απλή τροποποίηση στον αλγόριθμο αυτό είναι ένας constant (1/2) approximation algorithm ο οποίος στηρίζεται στο παρακάτω θεώρημα

Θεώρημα

Έστω S* = argmax{v(s), v(a)}, όπου S είναι η λύση που επιστρέφει ο greedy και a είναι το στοιχείο με την μεγαλύτερη density που όμως δεν ανήκει στο S. Τότε, $v(S^*) \geq \frac{OPT}{2}$, όπου OPT αποτελεί το value μιας βέλτιστης λύσης.

Απόδειξη

Έστω OPT_{LP} είναι το value μιας βέλτιστης λύσης για το fractional knapsack problem.

Τότε, $OPT \leq OPT_{LP}$ αφού το fractional knapsack problem είναι ένα relaxation του Knapsack.

Επίσης, είναι γνωστό από την κατασκευή του fractional optimal problem ότι $OPT_{LP} \leq v(S) + v(a)$ αφού μόνο ένα fraction $x_a \leq 1$ του τελευταίου στοιχείου προστίθεται.

Συνοψίζοντας, προκύπτει $OPT \leq OPT_{LP} \leq v(S) + v(a) \leq 2\max\{v(S), v(a)\}$

Dynamic Programming Algorithm για Knapsack

Let us define:

- $S_{i,p}$: subset of $\{a_1, \ldots, a_i\}$ with minimal cost which has value **exactly** p.
- $c(S_{i,p}) : \begin{cases} \infty & \text{if } S_{i,p} \text{ does not exist} \\ \sum_{a_i \in S_{i,p}} c(a_i) & \text{otherwise} \end{cases}$

and let us consider the following recurrence:

- $S_{1,p} = \begin{cases} \{a_1\} & \text{if } p = v_1 \\ \_ & \text{otherwise} \end{cases}$
- $S_{i+1,p} = \begin{cases} \text{argmin}\left[c(S_{i,p}), c(a_{i+1} \cup S_{i,p-v_{i+1}})\right] & \text{if } v_{i+1} \leq p \text{ and } S_{i,p-v_{i+1}} \neq \_ \\ S_{i,p} & \text{otherwise} \end{cases}$

Και ο αλγόριθμος θα είναι:

$$S_{1,p} = \begin{cases} a_1 & \text{if } p = v_i \\ \_ & \text{otherwise} \end{cases}$$

for $i \in \{1, \ldots, n\}$ **do**
  for $p \in \{1, \ldots, n \cdot V^*\}$ **do**

$$S_{i+1,p} = \begin{cases} \text{argmin}\left\{c(S_{i,p}), c(a_{i+1} \cup S_{i,p-v_{i+1}})\right\} & \text{if } v_{i+1} \leq p \text{ and } S_{i,p-v_{i+1}} \neq \_ \\ S_{i,p} & \text{otherwise} \end{cases}$$

  end for
end for
return $\text{argmax}_{p:c(S_{n,p}) \leq B} v(S_{n,p})$

Ο αλγόριθμος αυτός υπολογίζει μια βέλτιστη λύση για το Knapsack. Έχει πολυπλοκότητα $O(n^2V^*)$. Παρ' όλο που φαίνεται να κάνει πολυωνυμικό χρόνο, στην πραγματικότητα είναι pseudo-polynomial (αφού η πολυπλοκότητα μετριέται ως μια συνάρτηση μεγέθους ίσου με την είσοδο). Η τιμή V* γράφεται χρησιμοποιώντας logV* bits και για αυτό το λόγο, ένα polynomial-time algorithm πρέπει να έχει πολυπλοκότητα $O(\log^c V^*)$ για $c \geq 1$. Η πολυπλοκότητα $O(n^2V^*)$ είναι ουσιαστικά εκθετική μεγέθους ίσου με την είσοδο.

Polynomial-tome Approximation Scheme για Knapsack

Require: parameter ε
1: $k \leftarrow \frac{\varepsilon V^*}{n}$
2: **for** $i \in \{1, \ldots, n\}$ **do**
3: $\quad v_i' \leftarrow \lfloor \frac{v_i}{k} \rfloor$
4: **end for**
5: **return** the solution given by Algorithm 2 for input $\left(\{(c_1, v_1'), (c_2, v_2'), \ldots, (c_n, v_n')\}, B\right)$

Θεώρημα

Ο παραπάνω αλγόριθμος έχει πολυπλοκότητα $O(n^3/\varepsilon)$ και επιστρέφει μια λύση S τέτοια ώστε:

$v(S) \geq (1 - \varepsilon)OPT$

Απόδειξη

Running time: Όπως αναφέρεται και παραπάνω, η πολυπλοκότητα θα είναι: $O(n^2 a)$

$$O(n^2 \cdot \alpha) = O\left(n^2 \cdot \left\lfloor \frac{V^*}{k} \right\rfloor\right) = O\left(n^2 \cdot \frac{n}{\varepsilon}\right) = O\left(\frac{n^3}{\varepsilon}\right)$$

Approximation Ratio:

$$\begin{aligned} v(S) \geq k \cdot v'(S) &\geq k \cdot v'(O) \geq v(O) - k \cdot |O| \\ &\geq v(O) - k \cdot n \\ &= OPT - \varepsilon \cdot V^* \\ &\geq (1 - \varepsilon)OPT \end{aligned}$$

Resources:

1. https://people.seas.harvard.edu/~yaron/AM221-S16/lecture_notes/AM221_lecture17.pdf
2. https://ocw.mit.edu/courses/sloan-school-of-management/15-083j-integer-programming-and-combinatorial-optimization-fall-2009/lecture-notes/MIT15_083JF09_lec21.pdf
3. http://ac.informatik.uni-freiburg.de/lak_teaching/ws11_12/combopt/notes/knapsack.pdf
4. https://courses.engr.illinois.edu/cs598csc/sp2009/Lectures/lecture_4.pdf

c) Παράγραφοι 8.2 και 8.3 από Κεφάλαιο 8 Vazirani (σελ. 84 - 87)

Resources:

1. https://doc.lagout.org/science/0_Computer%20Science/2_Algorithms/Approximation%20Algorithms%20%5BVazirani%202010-12-01%5D.pdf

Βιβλιογραφία

1. *Διαφάνειες μαθήματος στο CoreLab*
2. *V.V. Vazirani. Approximation Algorithms: Chapters 2, 3*
3. *A. Levitin. Introduction to the Design and Analysis of Algorithms: Chapter 10.1*
4. *Michael X. Goemans, Linear Programming*
5. *Σχήματα Άσκησης 9: https://resources.mpi-inf.mpg.de/conferences/adfocs-15/material/David-Lect2.pdf?fbclid=IwAR1pE8TZ-Jl9Zea4kECeVhG13FukPvoF7Uwhi61sGbxhplCLWMKv3rPHjJU*