Προηγμένα Θέματα Αλγορίθμων
1η Σειρά Γραπτών Ασκήσεων

Γεώργιος Γκοτζιάς
03119047

Εαρινό εξάμηνο 2022-2023

Άσκηση 1

(α) **Αλγόριθμος**

Algorithm 1 Cardinality Set Cover

```
S ← ∅
C ← U
while C ≠ ∅ do                    ▷ Every element is covered when C is empty
    Choose e_i ∈ C
    for each S_j : e_i ∈ S_j do
        S ← S ∪ S_j
        for each e ∈ S_j do
            C ← C − {e}
        end for
    end for
end while
```

Ορθότητα

Το σύνολο S είναι Set Cover. γιατί όταν προστίθεται στο S το σύνολο S_j τα στοιχεία που ανήκουν σε αυτό, άρα αφαιρούνται από το C. Άρα, ο μόνος τρόπος να αφαιρεθεί ένα στοιχείο από το C είναι να καλύπτεται από κάποιο σύνολο του S. Ο αλγόριθμος τερματίζει, όταν είναι κενό το C, δηλαδή είναι όλα τα στοιχεία καλυμένα, άρα το S είναι Set Cover.
Ο αλγόριθμος τερματίζει πάντα, γιατί σε κάθε επανάληψη αφαιρείται από το C τουλάχιστον ένα στοιχείο (το e_i) και το U είναι πεπερασμένο. άρα το πολύ σε $|U|$ επαναλήψεις τερματίζει.

Λόγος προσέγγισης

Σε κάθε βήμα προστίθονται το πολύ f στοιχεία, αφού κανένα στοιχείο δεν ανήκει σε παραπάνω από f σύνολα. Από αυτά τα σύνολα τουλάχιστον 1 ανήκει στη βέλτιστη λύση, καθώς διαφορετικά δεν θα ήταν Set Cover, αφού το στοιχείο που επιλέξαμε δεν θα καλύπτοταν, αν δεν άνηκε κανένα. Άρα από τα OPT σύνολα που ανήκουν στη βέλτιση λύση, έχουμε επιλέξει το πολύ $f * OPT$ σύνολα, άρα ο αλγόριθμός μας είναι f-προσεγγιστικός.

Παρατήρηση

Ο παραπάνω αλγόριθμος δεν γενικεύει την έννοια του Maximal Matching άμεσα. Όμως, αν θεωρήσουμε για στοιχεία τις ακμές και ένα σύνολο για κάθε κόμβο με στοιχεία τις κορυφές που προσπίπτουν σε αυτόν, με την παραπάνω διαδικασία, όπου επιλέγουμε στοιχείο-ακμή και αποκλείουμε τα γειτονικά του, καταλήγουμε να πάρουμε ένα Maximal Matching σε περίπτωση που είχαμε Γράφο και καλούμασταν να υπολογίσουμε Vertex Cover.

Tight Example

Εφόσον δείξαμε ότι έχουμε γενίκευση του Vertex Cover, μπορούμε να κατασκευάσουμε έναν πλήρη διμερή γράφο και να θεωρήσουμε ως στοιχεία τις ακμές του και σύνολα τους κόμβους. Σε κάθε γράφο ισχύει $f = 2$, και όπως γνωρίζουμε σε αυτή την περίπτωση η λύση που υπολογίζουμε θα είναι ακριβώς η διπλάσια της βέλτιστης, επομένως έχουμε ένα tight example.

(β) Ο αλγόριθμος δεν είναι f-προσεγγιστικός για το πρόβλημα, όπως δεν είναι ο αντίστοιχος για το Vertex Cover. Μπορούμε να θεωρήσουμε το παρακάτω αντιπαράδειγμα:
$U = \{e_1, e_2, e_3, e4\}$
$S_1 = \{e_1, e_2\}$, με κόστος 1,
$S_2 = \{e_3, e_4\}$, με κόστος 1,
$S_3 = \{e_1, e_2, e_3, e_4\}$, με κόστος 50
Ο παραπάνω αλγόριθμος υπολογίζει ως λύση είτε την $\{S_1.S_3\}$ είτε την $\{S_2.S_3\}$, οι οποίες έχουν κόστος 51, όμως η βέλτιστη λύση είναι η $\{S_1, S_2\}$, η οποία έχει κόστος 2, άρα αν ήταν f-προσεγγιστικός ο αλγόριθμος θα έπρεπε να έβρισκε λύση με κόστος το πολύ 4, αφού $f = 2$.

(γ) **Αλγόριθμος**

Algorithm 2 Weighted Set Cover

```
C ← ∅
∀s ∈ S, t(s) ← cost(s)                              ▷ S = S1 ∪ S2 ∪ S3 ∪ ...
∀e ∈ U, c(e) ← 0
while C δεν είναι Set Cover do
    e ← ένα ακάλυπτο στοιχείο
    d ← ελάχιστο t(S) από τα S, για τα οποία e ∈ S
    for each S : e ∈ S do
        t(s) ← t(s) − d
    end for
    c(e) ← d
    C ← C ∪ {s ∈ S|t(s) = 0}
end while
```

Ορθότητα

Ο αλγόριθμος τερματίζει, όταν τερματίζει ο βρόχος while, δηλαδή αν το C είναι Set Cover. Άρα, για την απόδειξη της ορθότητας αρκεί να δειχθεί ότι ο αλγόριθμος τερματίζει. Πράγματι, σε κάθε επανάληψη προστίθεται τουλάχιστον ένα σύνολο, το οποίο περιέχει τουλάχιστον ένα ακάλυπτο στοιχείο (το στοιχείο e). Άρα, σε το πολύ $|U|$ επαναλήψεις ο αλγόριθμος τερματίζει.

Λόγος προσέγγισης

Στο i-οστό βήμα προσθέτουμε κόστος ίσο με $f * d_i$, όπου η ισότητα ισχύει αν στο τέλος έχουν επιλεχθεί όλα τα σύνολα που ανήκει το στοιχείο e_i που επιλέξαμε. Άρα, για το συνολικό κόστος ισχύει $\sum_{e \in U} c(e) \leq \sum_{i=1}^{k} f * d_i$, όπου k ο συνολικός αριθμός επαναλήψεων. Όμως, σε κάθε επανάληψη επιλέγεται ακάλυπτο στοιχείο, οπότε τουλάχιστον 1 από τα σύνολα ανήκουν στη βέλτιστη λύση, για να είναι έγκυρη. Άρα, το κόστος με το οποίο καλύπτεται ένα στοιχείο στη βέλτιστη λύση θα είναι τουλάχιστον ίσο με το d_i, που υπολογίζει ο αλγόριθμος, δηλαδή $OPT \geq \sum_{i=1}^{k} d_i$, οπότε $f * OPT \geq f * \sum_{i=1}^{k} d_i \geq \sum_{e \in U} c(e)$. Επομένως, η λύση που υπολογίζει ο αλγόριθμος είναι το πολύ f φορές η βέλτιστη.

Tight Example

Έστω το παρακάτω παράδειγμα:
$U = \{e_1, e_2, e_3\}$
$S_1 = \{e_1\}$, με κόστος 10
$S_2 = \{e_2\}$, με κόστος 10
$S_3 = \{e_3\}$, με κόστος 10
$S_4 = \{e_1, e_2, e_3\}$, με κόστος 30
Ο αλγόριθμος υπολογίζει για λύση την $\{S_1, S_2, S_3, S_4\}$, η οποία έχει συνολικό κόστος 60, όμως βέλτιστη λύση είναι η $\{S_1, S_2, S_3\}$ ή η $\{S_4\}$, η οποία έχει κόστος 30. Όμως, είναι $f = 2$, άρα η λύση που υπολογίζεται είναι φ φορές η βέλτιστη.
Μπορεί να γενικευτεί για tight example με $f = 2$, για κάθε πρόβλημα όπου έχουμε ένα μονοσύνολο για κάθε στοιχείο και ένα ακόμη σύνολο ίσο με το U, του οποίου το κόστος είναι ίσο με το άθροισμα όλων των προηγούμενων.

Άσκηση 2

Έστω το σύνολο $S = \{e_1, e_2, e_3, ..., e_{|S|}\}$, όπου τα στοιχεία του είναι αριθμημένα με τη σειρά που καλύπτονται. Αν επιλέξουμε το ίδιο το σύνολο το S, τότε καλύπτουμε όλα τα στοιχεία με κόστος $cost(S)$, άρα θα είναι $price(e_1) \leq \frac{cost(S)}{|S|}$, αφού ή θα επιλέξουμε το σύνολο S ή άλλο σύνολο που δίνει καλύτερη τιμή. Για το δεύτερο στοιχείο θα είναι $price(e_2) \leq \frac{cost(S)}{|S|-1}$, αφού η θα πάρουμε το κόστος του συνόλου S και θα έχουμε επιλέξει το e_1 μέσω κάποιου άλλου συνόλου, οπότε θα ισχύει η ισότητα ή θα πάρουμε κάποια καλύτερη λύση. Γενικά, για το k-οστό στοιχείο θα ισχύει $price(e_k) \leq \frac{cost(S)}{|S|-k+1}$(με επαγωγή). Επομένως, θα είναι $\sum_{k=1}^{|S|} price(e_k) \leq \sum_{k=1}^{|S|} \frac{cost(S)}{|S|-k+1} = H_{|S|} \cdot cost(S)$.
Έστω η βέλτιστη λύση $S_1, S_2, S_3, S_4, S_5, ..., S_k$, δηλαδή το βέλτιστο κόστος είναι

$OPT = cost(S_1) + cost(S_2) + cost(S_3) + ... + cost(S_k)$. Ο αλγόριθμος βρίσκει λύση με κόστος $SOL \leq \sum_{e \in S_1} price(e) + \sum_{e \in S_2} price(e) + ... + \sum_{e \in S_k} price(e)$, αφού $S_1 \cup S_2 \cup ... \cup S_k = U$, γιατί είναι Set Cover και κάποια στοιχεία πιθανόν να καλύπτονται παραπάνω από μία φορά, γι' αυτό και η λύση μπορεί να είναι μικρότερη. Όμως, $SOL \leq \sum_{e \in S_1} price(e) + \sum_{e \in S_2} price(e) + ... + \sum_{e \in S_k} price(e) \leq H_{|S_1|} \cdot cost(S_1) + H_{|S_2|} \cdot cost(S_2) + ... + H_{|S_k|} \cdot cost(S_k) \leq H_{|S_{max}|} \cdot (cost(S_1) + cost(S_2) + ... + cost(S_k)) = H_{|S_{max}|} \cdot OPT$.
Χρησιμοποιήθηκε ότι $H_{|S_i|} \leq H_{|S_{max}|}, \forall |S_i| \leq |S_{max}|$.
Άρα, ο αλγόριθμος είναι $H_{|S_{max}|}$-προσεγγιστικός.

Άσκηση 3

(α) Θεωρούμε ότι επιλέγεται σε κάθε επανάληψη το σύνολο με το μεγαλύτερο άθροισμα των βαρών των νέων (ακάλυπτων) στοιχείων.
Έστω S_{OPT} μία βέλτιστη λύση της οποίας το άθροισμα όλων των στοιχείων που καλύφθηκαν είναι W_{OPT}. Επειδή η λύση περιέχει ακριβώς k σύνολα υπάρχει σύνολο στην S, του οποίου τα στοιχεία έχουν συνολικό βάρος τουλάχιστον $\frac{W_{OPT}}{k}$.
Άρα, στο πρώτο βήμα επιλέγεται σύνολο με τουλάχιστον τόσο βάρος.
Επομένως, μένει ακάλυπτο το πολύ ένα μέρος $(1 - \frac{1}{k})$ των στοιχείων της βέλτιστης λύσης. Γενικά, σε κάθε βήμα το ακάλυπτο μέρος της βέλτιστης λύσης θα μειώνεται κατά τουλάχιστον $\frac{1}{k}$, οπότε στην i-οστή επανάληψη ακάλυπτο θα είναι το πολύ ένα μέρος $(1 - \frac{1}{k})^i$ της βέλτιστης λύσης.
Άρα, ο Greedy αλγόριθμος υπολογίζει λύση $SOL \geq (1 - (1 - \frac{1}{k})^k) \cdot W_{OPT} > (1 - \frac{1}{e}) \cdot W_{OPT}$.

(β) Θεωρούμε ως κριτήριο επιλογής συνόλου σε κάθε επανάληψη τον μέγιστο λόγο $\frac{|S-C|}{w(S)}$, όπου S σύνολο που δεν έχει επιλεγεί, C το σύνολο των στοιχείων που έχουν καλυφθεί μέχρι εκείνο το σημείο και $w(S)$ το βάρος του συνόλου S.
Ο αλγόριθμος έχει άνω φράγμα για τον λόγο προσέγγισης $\frac{1}{2}$, όπως διαπιστώνουμε από το παρακάτω παράδειγμα:
Έστω σύνολα με βάρη $(\frac{B}{2} + \varepsilon)$, $\frac{B}{2}$ και $\frac{B}{2}$, τα οποία καλύπτουν $n+1$, n και n στοιχεία αντίστοιχα, όλα διαφορετικά μεταξύ τους. Ο παραπάνω αλγόριθμος θα επέλεγε μόνο το πρώτο, εφόσον $\varepsilon < \frac{B}{2n}$, καθώς θα ίσχυε $\frac{n+1}{\frac{B}{2}+\varepsilon} > \frac{n}{\frac{B}{2}}$. Δεν θα επιλέγοταν κάποιο άλλο σύνολο, αφού θα υπολείποταν budget $(\frac{B}{2} - \varepsilon)$, άρα θα καλύπτονταν συνολικά $(n+1)$ στοχεία. Αν είχαμε επιλέξει τα άλλα δύο σύνολα όμως θα είχαν καλυφθεί $2n$ στοιχεία, οπότε έχουμε $\frac{SOL}{OPT} = \frac{n+1}{2n}$, το οποίο για μεγάλα n τείνει στο $\frac{1}{2}$, οπότε δεν ισχύει σίγουρα ο λόγος προσέγγισης που υπάρχει για τις άλλες δύο παραλλαγές του προβλήματος.
Αν θεωρήσουμε ότι τα βάρη δεν είναι ακέραια, τότε δεν υπάρχει καμία θετική σταθερά που να φράσει τον λόγος προσέγγισης. Έστω το παρακάτω παράδειγμα:
Έστω σύνολο με βάρος $\frac{1}{N}$, όπου N ακέραιος, με 1 στοιχείο και σύνολο με βάρος B, το οποίο καλύπτει $(N-1) \cdot B$ στοιχεία (Θεωρούμε ότι B ακέραιος). Ο αλγόριθμος μας υπολογίζει ως βέλτιση λύση να επιλεχθεί μόνο το πρώτο σύνολο,

αφού έχει λόγο $\frac{1}{\frac{1}{N}} = N$, ενώ το άλλο έχει βάρος $\frac{(N-1)\cdot B}{B} = N-1$. Στη συνέχεια, δεν μπορεί να επιλεχθεί και το δεύτερο σύνολο, καθώς δεν επαρκεί το budget. Όμως, βέλτιστη λύση θα είναι μόνο το δεύτερο σύνολο, επομένως ο λόγος προσέγγισης θα είναι $\frac{SOL}{OPT} = \frac{1}{(N-1)\cdot B}$, οπότε για οποιαδήποτε θετική σταθερά ε για το λόγο προσέγγισης μπορούμε να βρούμε παράδειγμα με ακόμα μικρότερο λόγο προσέγγισης.

Άσκηση 4

Αναγωγή διατήρησης παράγοντα προσέγγισης από το Dominating Set στο Set Cover

Έστω γράφος $G(V, E)$. Ορίζουμε το σύνολο $U = V$ και για κάθε κορυφή v_i στο V δημιουργούμε το σύνολο S_i, το οποίο περιέχει την κορυφή v_i και κάθε γειτονική της κορυφή. Το πρόβλημα Dominating Set για τον γράφο G ανάγεται πολυωνυμικά στο πρόβλημα Set Cover για τα σύνολα U, $S = \{S_1, S_2, ..\}$.

Έστω ότι γνωρίζουμε τη βέλτιστη λύση για το πρόβλημα Dominating Set, τότε η επιλογή των αντίστοιχων συνόλων S_i, για κάθε κόμβο στο D είναι λύση και για το Set Cover, καθώς κάθε στοιχείο v_i ή θα ανήκει στο D, άρα θα έχει επιλεγεί στο σύνολο S_i, ή θα έχει μία γειτονική κορυφή v_j στο D, άρα θα ανήκει στο σύνολο S_j. Σε κάθε περίπτωση, κάθε στοιχείο του U είναι καλυμένο, άρα έχουμε αποδεκτή λύση για το Set Cover. Επομένως, ισχύει $OPT_{SetCover} \leq OPT_{DominatingSet}$.

Αν έχουμε μία λύση $S' = \{S_{i_1}, S_{i_2}, S_{i_3}, ...\}$ για το Set Cover, τότε θεωρούμε το σύνολο $D = \{v_{i_1}, v_{i_2}, v_{i_3}, ...\}$. Κάθε κορυφή του $V \setminus D$, θα είναι καλυμένη στο πρόβλημα Set Cover μέσω κάποιου συνόλου S_j, όμως για να ανήκει στο S_j σημαίνει ότι έχει γείτονική κορυφή την v_j, η οποία ανήκει στο D. Οπότε, η για κάθε λύση του Set Cover βρίσκουμε λύση για το Dominating Set ίδιου πλήθους, δηλαδή $SOL_{DominatingSet} \leq SOL_{SetCover}$. Επομένως, για κάθε ρ-προσεγγιστικό αλγόριθμο για το Set Cover υπάρχει ρ-προσεγγιστικός αλγόριθμος για το Dominating Set. Άρα, κάθε άνω φράγμα για το Set Cover είναι και άνω φράγμα για το Dominating Set.

Αναγωγή διατήρησης παράγοντα προσέγγισης από το Set Cover στο Dominating Set

Έστω ότι δίνεται σύνολο $U = \{e_1, e_2, ..., e_n\}$ και σύνολο συνόλων
$S = \{S_1, S_2, S_3, ..., S_m\}$, όπου $\forall i \in \{1, 2, 3, ..., m\}, S_i \subseteq U$.

Ορίζουμε γράφο $G(V, E)$ με $V = \{e_1, e_2, e_3, ..., e_n, i_1, i_2, i_3, ..., i_m\}$ και ακμές $(i_k, i_j), \forall k, j \in \{1, 2, 3, ..., m\}, k \neq j$, καθώς και ακμές $(e_k, i_j), \forall k, j$ τέτοια, ώστε $e_k \in S_j$. Έστω ότι γνωρίζουμε μία βέλτιστη λύση για το πρόβλημα Set Cover, την $S' = \{S_{j_1}, S_{j_2}, S_{j_3}, ...\}$, τότε το σύνολο $D = \{i_{j_1}, i_{j_2}, i_{j_3}, ...\}$ είναι αποδεκτή λύση για το πρόβλημα Dominating Set, γιατί κάθε κορυφή i_j, γειτονεύει με όλες τις κορυφές i_k, οπότε θα υπάρχει κάποια στο D, ενώ για κάθε κορυφή e_j το αντίστοιχο στοιχείο στο πρόβλημα Set Cover καλύπτεται από κάποιο σύνολο S_k, άρα έχει γείτονα την κορυφή i_k, όμως $S_k \in S'$, άρα $i_k \in D$. Οπότε έχουμε ότι $OPT_{DominatingSet} \leq OPT_{SetCover}$. Έστω ότι έχουμε μία λύση για το πρόβλημα Dominating Set. Υποθέτουμε χωρίς βλάβη της γενικότητας ότι η λύση έχει τη

μορφή $D = \{i_{i_1}, i_{i_2}, i_{i_3}, ...\}$. Αυτό γίνεται, γιατί οι κόμβοι i σχηματίζουν κλίκα μεταξύ τους, ενώ οι κόμβοι e ανεξάρτητο σύνολο, οπότε για οποιαδήποτε λύση που το D περιέχει στοιχεία e, μπορούμε να τα αντικαταστήσουμε με οποιοδήποτε γειτονικό τους στοιχείο, το οποίο θα είναι i, αφού το e που τώρα θα ανήκει στο $V \setminus D$, θα έχει το γείτονα που επιλέχθηκε στο D, ενώ κάθε άλλος γείτονας του θα έχει γείτονα και το i που επιλέχθηκε, άρα η λύση εξακολουθεί να είναι έγκυρη.
Η επιλογή του συνόλου $S'' = \{S_{i_1}, S_{i_2}, S_{i_3}, ..\}$, θα είναι έγκυρη λύση για το πρόβλημα Set Cover, αφού κάθε κορυφή e_j, στο πρόβλημα Dominating Set έχει γείτονα κάποια κορυφή i_k, άρα το σύνολο $S_k \in S''$ θα καλύπτει το στοιχείο e_j.
Οπότε, έχουμε $SOL_{SetCover} \leq SOL_{DominatingSet}$.
Η αναγωγή αποδείχθηκε, επομένως αν υπάρχει ρ-προσεγγιστικός αλγόριθμος για το πρόβλημα Dominating Set θα υπάρχει και ρ-προσεγγιστικός αλγόριθμος για το πρόβλημα Set Cover, άρα τα κάτω φράγματα που ισχύουν για το Set Cover ισχύουν και για το Dominating Set.

Άσκηση 5

Για το SOL_1:
Είναι $SOL_1 \leq cost(P_{s,t})(1)$,
καθώς ισχύει η τριγωνική ανισότητα, οπότε με short-cutting, το κόστος θα μειώνεται. Ακόμη, ισχύει $cost(P_{s,t}) \leq 2 \cdot cost(T) - c_{s,t}(2)$,
γιατί ο γράφος που υπολογίζουμε το μονοπάτι Euler αποτελείται από το Minimum Spanning Tree T με διπλασιασμένες τις ακμές(άρα $cost(P_{s,t}) \leq 2 \cdot cost(T)$), μετά από την αφαίρεση του μονοπατιού (s, t).
Το βέλτιστο μονοπάτι Hamilton είναι Spanning Tree για τον αρχικό γράφο, οπότε θα ισχύει $cost(T) \leq OPT(3)$.
Επομένως, συνδυάζοντας τις σχέσεις $(1), (2), (3)$ παίρνουμε ότι $SOL_1 \leq 2 \cdot OPT_{s,t} - c_{s,t}$.
Ο όρος $c_{s,t}$ εκφράζει την αφαίρεση του μονοπατιού (s, t) πριν τον υπολογισμό του κύκλου Euler. (Λόγω τριγωνικής ανισότητας, το κόστος του μονοπατιού θα είναι τουλάχιστον $c_{s,t}$)
Για το SOL_2:
Είναι $SOL_2 \leq cost(P_{s,t})(4)$,
καθώς ισχύει η τριγωνική ανισότητα, οπότε με short-cutting, το κόστος θα μειώνεται. Ακόμη, ισχύει $cost(P_{s,t}) \leq cost(T) + cost(M)(5)$,
αφού υπολογίζουμε το Euler Path στον γράφο που αποτελείται από το Minimum Spanning Tree T και το Matching M.
Το βέλτιστο μονοπάτι Hamilton είναι Spanning Tree για τον αρχικό γράφο, οπότε θα ισχύει $cost(T) \leq OPT(6)$.
Στους κόμβους περιττού βαθμού του T παίρνουμε κύκλο με κόστος το πολύ $OPT + c_{s,t}$, γιατί διαφορετικά δεν θα υπήρχε λύση OPT για το πρόβλημα, αφού θα έπρεπε να πάρουμε έναν κύκλο με μεγαλύτερου κόστους εφόσον προσθέταμε κορυφές, λόγω τριγωνικής ανισότητας. Άρα, ισχύει $cost(M) \leq \frac{OPT + c_{s,t}}{2}(7)$.
Από τις σχέσεις $(5), (6), (7)$, προκύπτει $cost(P_{s,t}) \leq OPT + \frac{OPT + c_{s,t}}{2}$, το οποίο

αν συνδυαστεί με τη σχέση (4) παίρνουμε $SOL_2 \leq \frac{3 \cdot OPT_{s,t} + c_{s,t}}{2}$.
Ο όρος $c_{s,t}$ εκφράζει ότι η ακμή (s,t) μαζί με το μονοπάτι Euler, που υπολογίζεται, δίνουν έναν κύκλο Euler, ο οποίος θα έχει μικρότερο κόστος από το Matching.
Για το λόγο $\frac{5}{3}$:
Έστω ότι $SOL_1 > \frac{5}{3} \cdot OPT$, τότε θα είναι $2 \cdot OPT - c_{s,t} > \frac{5}{3} \cdot OPT \Rightarrow c_{s,t} < \frac{OPT}{3}$.
Οπότε θα είναι $SOL_2 \leq \frac{3 \cdot OPT_{s,t} + c_{s,t}}{2} < \frac{3 \cdot OPT_{s,t} + \frac{OPT}{3}}{2} = \frac{5}{3} \cdot OPT$, οπότε θα είναι $SOL < \frac{5}{3} \cdot OPT$, αφού $SOL = min(SOL_1, SOL_2)$.
Έστω ότι $SOL_2 > \frac{5}{3} \cdot OPT$, οπότε θα είναι $\frac{3 \cdot OPT + c)s,t}{2} > \frac{5}{3} \cdot OPT \Rightarrow c_{s,t} > \frac{OPT}{3}$.
Επομένως, έχουμε $SOL_1 \leq 2 \cdot OPT - c_{s,t} < 2 \cdot OPT - \frac{OPT}{3} = \frac{5}{3} \cdot OPT$, άρα θα είναι και $SOL < \frac{5}{3} \cdot OPT$. Επομένως, σε κάθε περίπτωση ο λόγος προσέγγισης είναι $\frac{5}{3}$.

Άσκηση 6

Έστω ότι οι χρόνοι ολοκλήρωσης για κάθε εργασία είναι $p_1 > p_2 > ... > p_n$. Αν OPT η βέλτιστη λύση, χωρίζουμε τις εργασίες σε ¨μεγάλες¨, για τις οποίες ισχύει $p_i > \frac{OPT}{3}$ και σε ¨μικρές¨, τις υπόλοιπες. Υπάρχουν, το πολύ $2m$ ¨μεγάλες' εργασίες, καθώς αν θεωρήσουμε ότι υπάρχουν τουλάχιστον $2m + 1$, τότε, από αρχή του περιστερώνα, θα έπρεπε να τοποθετηθούν, τουλάχιστον, 3 ¨μεγάλες' εργασίες στην ίδια μηχανή. Όμως, αυτή η μηχανή θα είχε χρονο ολοκλήρωσης $> 3 \cdot \frac{OPT}{3} = OPT$, το οποίο είναι άτοπο.
Θα δείξουμε ότι όλες οι ¨μεγάλες' εργασίες έχουν ολοκληρωθεί πριν τον χρόνο OPT. Αν θεωρήσουμε ότι έχουμε ακριβώς $2m$ ¨μεγάλες' εργασίες, τότε στη μηχανή 1, θα τοποθετηθούν οι εργασίες p_1, p_{2m}, στη μηχανή 2 οι εργασίες p_2, p_{2m-1} και γενικά στη μηχανή i οι εργασίες p_i, p_{2m-i+1}. Έστω, ότι στη μηχανή k, οι μεγάλες εργασίες ολοκληρώνονται σε χρόνο $> OPT$, δηλαδή $p_k + p_{2m-k+1}$. Αυτό είναι άτοπο, γιατί αν ίσχυε κάτι τέτοιο, δεν θα υπήρχε λύση OPT, γιατί αν προσπαθήσουμε να ανταλλάξουμε τις δεύτερες εργασίες ανάμεσα στη μηχαμή k και μία μηχανή j, αν $j < k$ τότε η μηχανή j θα είχε χρόνο ολοκλήρωσης $p_j + p_{2m-k+1} > p_k + p_{2m-k+1} > OPT$, ενώ αν $j > k$ η μηχανή k μετά την αλλαγή θα είχε χρόνο ολοκλήρωσης $p_k + p_{2m-j+1} > p_k + p_{2m-k+1} > OPT$.
Κάθε εργασία θα έχει χρόνο έναρξης $< OPT$, γιατί διαφορετικά θα έπρεπε όλες οι μηχανές να είναι απασχολημένες για όλο τον χρόνο OPT, όμως τότε δεν θα υπήρχε βέλτιστη λύση ίση με OPT.
Η εργασίες που μπορούν να τερματίσουν μετά από τον χρόνο OPT θα είναι ¨μικρές¨, γιατί όπως δείξαμε δεν γίνεται οι μεγάλες να τερματίσουν μετά από αυτόν τον χρόνο, άρα η λύση που υπολογίζεται είναι $SOL \leq OPT + \frac{OPT}{3} = \frac{4}{3} \cdot OPT$, άρα ο λόγος προσέγγισης είναι $\frac{4}{3}$.

Άσκηση 7

(α) Ο λόγος προσέγγισης δεν φράσσεται από καμία σταθερά
Έστω το παρακάτω παράδειγμα:
Έχουμε δύο αντικείμενα, με το πρώτο να έχει βάρος $\frac{1}{N}$ και αξία 1, ενώ το δεύτερο έχει βάρος B και αξία $(N-1)\cdot B$. Ο αλγόριθμος επιλέγει το πρώτο αντικείμενο αφού $\frac{p(1)}{w(1)} = \frac{1}{\frac{1}{N}} = N$, ενώ το δεύτερο έχει $\frac{p(2)}{w(2)} = \frac{(N-1)\cdot B}{B} = N-1$. Στη συνέχεια ο αλγόριθμος τερματίζει, αφού δεν χωράει και το δεύτερο στοιχείο στη λύση. Οπότε ο λόγος προσέγγισης είναι $\frac{SOL}{OPT} = \frac{1}{(N-1)\cdot B}$, και επειδή το N είναι αυθαίρετο δεν υπάρχει σταθερά που να φράσσει τον λόγο.

(β) Ο λόγος προσέγγισης είναι $\frac{1}{2}$
Έστω $p_1, p_2, ..., p_n$ με $\frac{p_1}{w_1} > \frac{p_2}{w_2} > ... > \frac{p_n}{w_n}$. Αν k είναι το πρώτο στοιχείο που δεν επιλέγεται, τότε θα ισχύει $p_1+p_2+...+p_{k-1}+p_k \geq OPT$, γιατί $p_1+p_2+...+p_{k-1}+p_k \geq p_1+p_2+...+p_{k-1}+a\cdot p_k$, όπου $a \in [0,1]$, όμως $p_1+p_2+...+p_{k-1}+a\cdot p_k$, για συγκεκριμένο a είναι η λύση για το Fractional Knapsack, η οποία είναι καλύτερη (ή το ίδιο καλή) με τη λύση του Discrete Knapsack, καθώς κάθε λύση του δεύτερου είναι αποδεκτή λύση του πρώτου. Επομένως, ισχύει $p_1+p_2+...+p_{k-1}+p_k \geq OPT$, οπότε είναι $p_1+p_2+...+p_{k-1} \geq \frac{OPT}{2}$ ή $p_k \geq \frac{OPT}{2}$. Η λύση που επιστρέφει ο αλγόριθμος θα είναι η $p_1+p_2+...+p_{k-1}$ ή κάποιο p_j τέτοιο, ώστε $p_j > p_k$, άρα σε κάθε περίπτωση $SOL \geq \frac{OPT}{2}$.

(γ) PTAS Αλγόριθμος
Αντί να δοκιμάζουμε το στοιχείο με τη μεγαλύτερη αξία πρέπει να δοκιμάζουμε όλα τα δυνατά (που είναι της τάξης του $O(n^k)$ υποσύνολα με πληθάριθμο το πολύ k και στη συνέχεια συμπληρώνουμε με Greedy κριτήριο.