Greedy Algorithms

Αδάμος Τοφαρή

Περιεχόμενα

- Εισαγωγή
- Coin Problem
  - Ο άπληστος αλγόριθμος
  - Ορθότητα
  - Γενική Περίπτωση
- Scheduling
- Tasks and Deadlines

Εισαγωγή

Ένας άπληστος αλγόριθμος πάντα χτίζει την λύση κάνοντας την επιλογή που συμφέρει περισσότερο την συγκεκριμένη στιγμή Ο άπληστος αλγόριθμος ποτέ δεν παίρνει πίσω την απόφαση του, αλλά χτίζει απευθείας την τελική λύση Για αυτό τον λόγο, οι άπληστοι αλγόριθμοι είναι συνήθως πολύ αποτελεσματικοί.

Η δυσκολία στην σχεδίαση ενός άπληστου αλγόριθμου είναι η εύρεση άπληστης στρατηγικής η οποία παράγει την βέλτιστη λύση του προβλήματος. ΟΙ τοπικές βέλτιστες επιλογές σε ένα άπληστο αλγόριθμο πρέπει να είναι και καθολικά βέλτιστες. Συνήθως είναι δύσκολο να αμφισβητήσουμε ότι ένας άπληστος αλγόριθμος δουλεύει.

Coin problem

Ως πρώτο παράδειγμα, ας σκεφτούμε ένα πρόβλημα που μας δίνετε ένα σύνολο από κέρματα και το έργο μας είναι να παράγουμε το άθροισμα χρημάτων n με τη χρήση των κερμάτων. Οι τιμές των κερμάτων είναι coins=$\{c_1, c_2, \dots, c_k\}$ και κάθε κέρμα μπορούμε να χρησιμοποιήσουμε όσες φορές χρειαζόμαστε. Ποιος είναι ο ελάχιστος αριθμός κερμάτων που χρειαζόμαστε;

Για παράδειγμα, αν τα κέρματα είναι cents του Euro:

{1,2,5,10,20,50,100,200}

και x=520, χρειαζόμαστε τουλάχιστον 4 κέρματα. Η βελτίστη λύση είναι να επιλέξουμε τα κέρματα 200+200+100+20 των οποίων το άθροισμα είναι 520.

Ο άπληστος αλγόριθμος

Ένας απλός άπληστος αλγόριθμος για το πρόβλημα πάντα επιλέγει το μεγαλύτερο δυνατόν κέρμα, μέχρι το ζητούμενο άθροισμα λεφτών έχει παραχθεί. Αυτός ο αλγόριθμος μας βοήθα στο παράδειγμα γιατί πρώτα επιλέγει 2 των 200 cent κερμάτων, και μετά των 100 κερμάτων και τελικά ένα των 20 κερμάτων. Αλλά ο αλγόριθμος πάντα βγάζει το επιθυμητό αποτέλεσμα;

Αποδεικνύεται ότι για τα κέρματα των Euro, ο άπληστος αλγόριθμος *πάντα* λειτουργά, αυτό είναι, πάντα παράγει λύση με το ελάχιστο αριθμό κερμάτων.

Ορθότητα

Η ορθότητα του αλγόριθμου μπορεί να αποδειχθεί πιο κάτω:

Για κάθε κέρμα 1, 5, 10, 50 και 100 εμφανίζονται το πολύ μια φορά στην βέλτιστη λύση, γιατί αν η λύση περιέχει 2 τέτοια κέρματα, θα μπορούσαμε να τα αντικαταστήσουμε με ένα τέτοιο κέρμα και θα μπορούσαμε να πάρουμε καλύτερη λύση Για παράδειγμα, αν η λύση περιέχει κέρματα 5+5, θα μπορούσαμε να τα αντικαταστήσουμε με το κέρμα 10.

Με τον ίδιο τρόπο, τα κέρματα 2 και 20 εμφανίζονται το πολύ 2 φορές στην βέλτιστη λύση, γιατί μπορούμε να αντικαταστήσουμε τα κέρματα 2+2+2 με τα κέρματα 5+1 και τα κέρματα 20+20+20 με τα κέρματα 50+10. Έξαλλου, η βέλτιστη λύση δεν μπορεί να περιέχει τα κέρματα 2+2+1 ή 20+20+10, επειδή μπορούμε να τα αντικαταστήσουμε με τα κέρματα 5 και 50.

Ορθότητα

Χρησιμοποιώντας αυτές τις παρατηρήσεις μπορούμε να δείξουμε ότι για κάθε κέρμα x ότι είναι αδύνατο να να παράγουμε το άθροισμα x ή οποιοδήποτε άλλο μεγαλύτερο άθροισμα με τη χρήση κερμάτων μικρότερων του x. Για παράδειγμα, αν x=100, το μεγαλύτερο βέλτιστο άθροισμα χρησιμοποιώντας τα μικρότερα κέρματα είναι 50+20+20+5+2+2=99. Έτσι, ο άπληστος αλγόριθμος ο οποίος πάντα επιλέγει το μεγαλύτερο κέρμα παράγει την βέλτιστη λύση.

Αυτό το παράδειγμα δείχνει ότι είναι δύσκολο να αμφισβητήσουμε ότι ο αλγόριθμος λειτουργεί, ακόμη και αν ο αλγόριθμος είναι απλός από μονός του.

Γενική Περίπτωση

Στην γενική περίπτωση, το σύνολο των κερμάτων περιέχει οποιαδήποτε κέρματα και ο άπληστος αλγόριθμος δεν παράγει απαραίτητα την βέλτιστη λύση

Μπορούμε να το αποδείξουμε ότι ο άπληστος αλγόριθμος δεν λειτουργά με το να δείξουμε αντιπαράδειγμα οπού ο αλγόριθμος δίνει λάθος απάντηση. Σε αυτό το πρόβλημα είναι εύκολο να δείξουμε αντιπαράδειγμα: αν τα κέρματα είναι {1,3,4} και το επιθυμητό άθροισμα είναι 6, ο άπληστος αλγόριθμος παράγει την λύση 4+1+1 ενώ η βέλτιστη λύση είναι 3+3.

Δεν είναι γνωστό αν η γενική περίπτωση του προβλήματος των κερμάτων λύνεται με κάποιο άπληστο αλγόριθμο. Ωστόσο, όπως θα δούμε στο Κεφάλαιο του Δυναμικού Προγραμματισμού, σε κάποιες περιπτώσεις, οι γενική περίπτωση του προβλήματος μπορεί να λυθεί αποτελεσματικά με την χρήση δυναμικού προγραμματισμού αλγόριθμος που θα δίνει πάντα ορθή απάντηση.

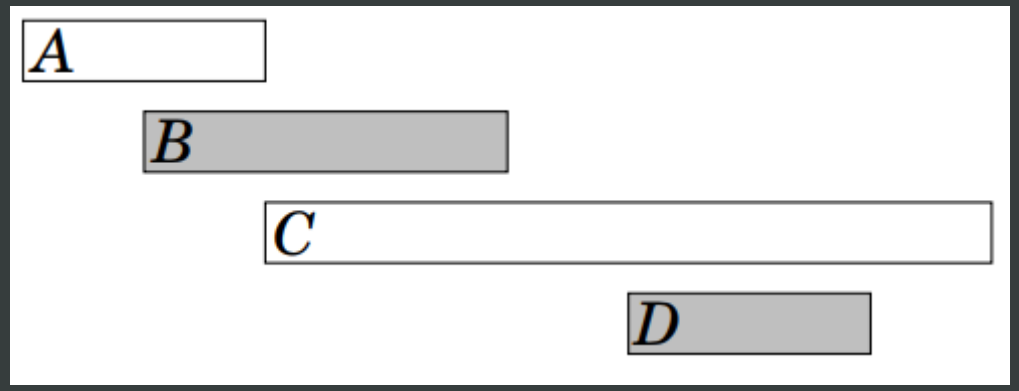
Scheduling

Πληθώρα προβλήματα Χρονοδρομολόγησης μπορούν να λυθούν με τη χρήση άπληστου αλγόριθμου. Ένα κλασικό πρόβλημα είναι το έξης: Δίνονται n εκδηλώσεις με τις χρονικές στιγμές που ξεκινούν και τελειώνουν, βρείτε ένα πρόγραμμα που να περιέχει τις περισσότερες εκδηλώσεις. Δεν είναι δυνατόν να επιλέξουμε μερική εκδήλωση. Για παράδειγμα, σκεφτείτε τις επόμενες εκδηλώσεις:

event	starting time	ending time
A	1	3
B	2	5
C	3	9
D	6	8

Scheduling

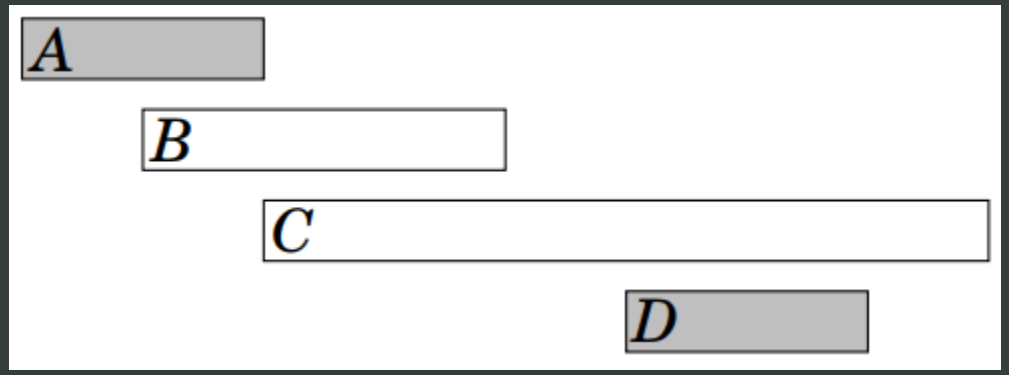
Σε αυτή την περίπτωση ο μέγιστος αριθμός εκδηλώσεων είναι 2. Για παράδειγμα μπορούμε να επιλέξουμε τις εκδηλώσεις B και D ως έξης:



Είναι δυνατόν να εφεύρουμε πολλαπλούς άπληστους αλγόριθμους, άλλα ποιος από αυτούς λειτουργά για κάθε περίπτωση;

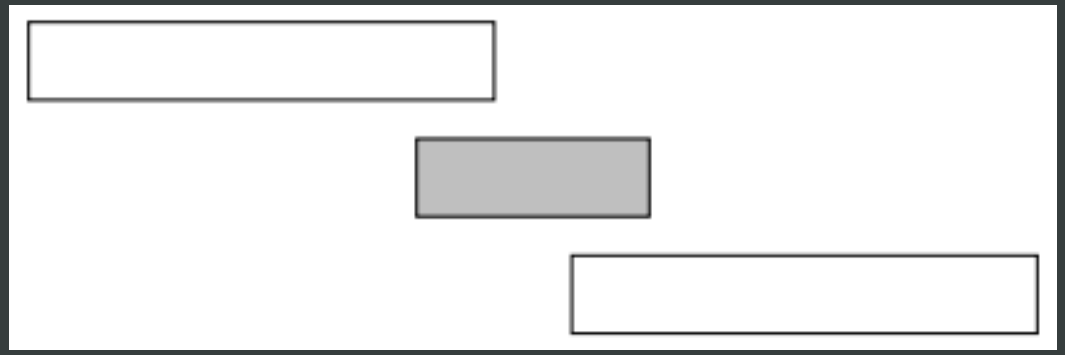
Αλγόριθμος 1

Η πρώτη ιδέα είναι να επιλέξουμε τις περισσότερες συντομότερες εκδηλώσεις. Στο παράδειγμα ο αλγόριθμος επιλέγει τις επόμενες εκδηλώσεις:



Αλγόριθμος 1

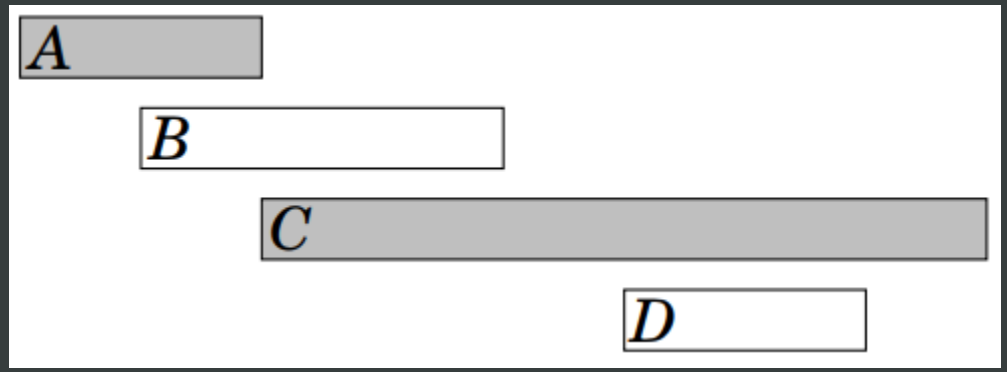
Ωστόσο, επιλέγοντας τις συντομότερες εκδηλώσεις δεν είναι πάντα η ορθή στρατηγική. Για παράδειγμα, ο αλγόριθμος αποτυγχάνει στη επομένη περίπτωση:



Αν επιλέξουμε τη σύντομη εκδήλωση, μπορούμε να επιλέξουμε μια εκδήλωση. Ωστόσο είναι δυνατόν να επιλέξουμε τις δυο μακρόχρονες εκδηλώσεις.

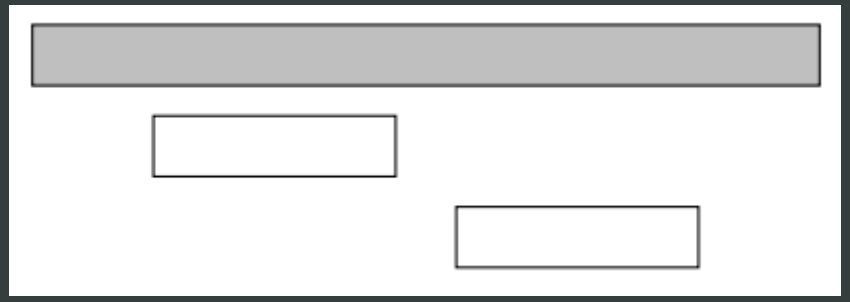
Αλγόριθμος 2

Άλλη ιδέα είναι να επιλέγουμε την επομένη δυνατή εκδήλωση η οποία ξεκινά όσο νωρίς γίνετε. Αυτός ο αλγόριθμος επιλέγει τις έξης εκδηλώσεις:



Αλγόριθμος 2

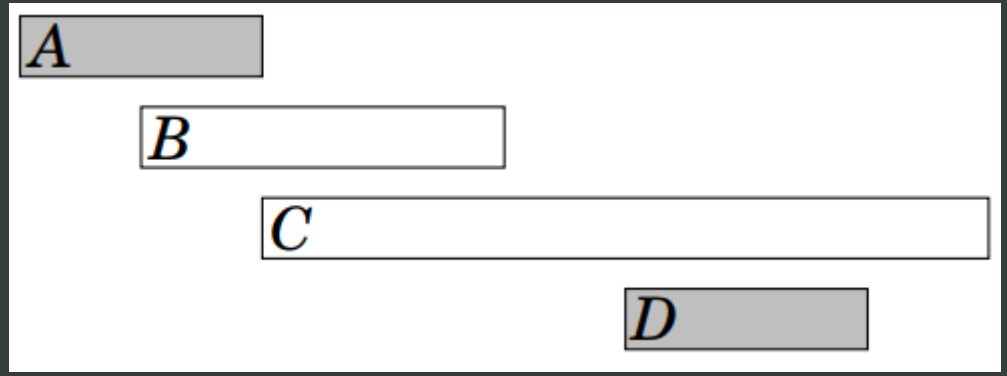
Ωστόσο, μπορούμε να βρούμε αντιπαράδειγμα και για αυτό τον αλγόριθμο. Για παράδειγμα, στο επόμενο παράδειγμα, ο αλγόριθμος επιλέγει μια εκδήλωση:



Αν επιλέξουμε την 1η εκδήλωση, δεν είναι δυνατόν να επιλέξουμε άλλες εκδηλώσεις. Ωστόσο, θα ήταν δυνατόν να επιλέξουμε τις άλλες 2 εκδηλώσεις.

Αλγόριθμος 3

Η 3η ιδέα είναι πάντα να επιλέγουμε την επομένη πιθανή εκδήλωση που τελειώνει όσο πιο νωρίς γίνετε Αυτός ο αλγόριθμος επιλέγει τις επόμενες εκδηλώσεις:



Αλγόριθμος 3

Αποδεικνύετε ότι ο αλγόριθμος πάντα παράγει την βέλτιστη λύση Ο λόγος για αυτό είναι ότι πάντα η βέλτιστη επιλογή είναι να επιλέξουμε πρώτα την εκδήλωση η οποία τελειώνει όσο πιο δυνατό νωρίς. Μετά από αυτό, είναι η βέλτιστη επιλογή να επιλέξουμε την επομένη εκδήλωση χρησιμοποιώντας την ίδια στρατηγική, κτλ., μέχρι δεν θα μπορέσουμε να επιλέξουμε άλλες εκδηλώσεις.

Ένας τρόπος να υποστηρίξουμε ότι ο αλγόριθμος λειτουργεί είναι να εξεταστεί τι συμβαίνει αν εμείς πρώτα επιλέξουμε μια εκδήλωση που τελειώνει αργότερα από την εκδήλωση που τελειώνει το συντομότερο δυνατόν. Τώρα θα έχουμε το πολύ ίσο αριθμό επίλογων για το πώς μπορούμε να επιλέξουμε τη επόμενη εκδήλωση. Επόμενος, η επιλογή ενός συμβάντος που τελειώνει αργότερα δεν μπορεί ποτέ να δώσει μια καλύτερη λύση, και ο άπληστος αλγόριθμος είναι σωστός.

Tasks and Deadlines

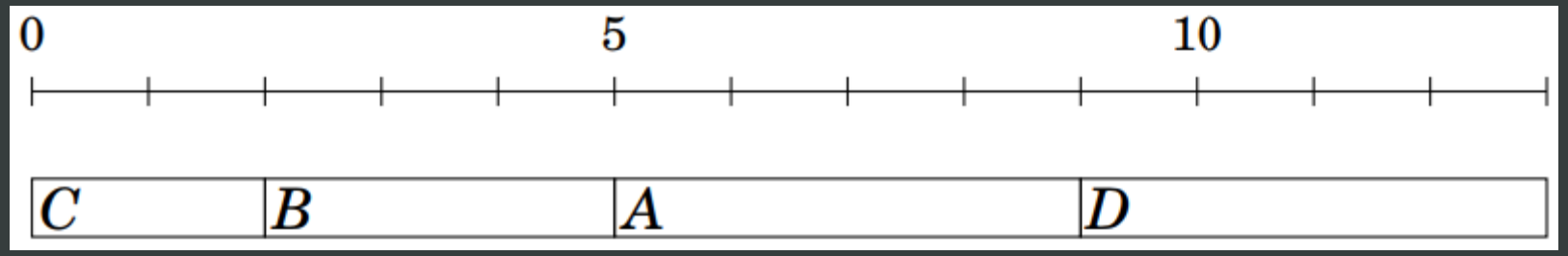
Τώρα ας σκεφτούμε ένα πρόβλημα οπού δίνονται n διεργασίες με διάρκεια και προθεσμία και το έργο μας είναι να επιλέξουμε την σειρά με την οποία θα εκπληρωθούν οι διεργασίες. Για κάθε διεργασία παίρνουμε $d - x$ πόντους οπού d η προθεσμία της διεργασίας και x η στιγμή που την ολοκληρώσαμε. Το ζητούμενο είναι ο μέγιστος αριθμός πόντων που μπορούμε να μαζέψουμε.

Για παράδειγμα, σκεφτείτε ότι οι διεργασίες είναι οι επόμενες:

task	duration	deadline
A	4	2
B	3	5
C	2	7
D	4	5

Tasks and Deadlines

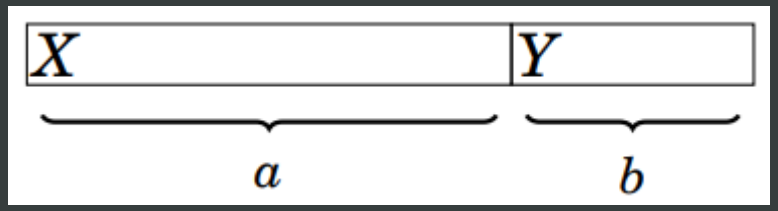
Σε αυτή τη περίπτωση, το βέλτιστο χρονοδιάγραμμα για τις διεργασίες είναι το επόμενο:



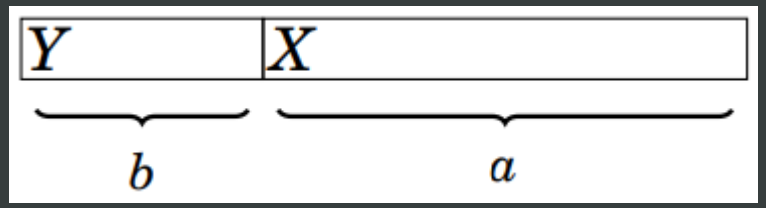
Σε αυτή τη λύση, C δίνει 5 πόντους, B δίνει 0 πόντους, A δίνει -7 πόντους και D δίνει -8 πόντους, έτσι οι συνολικοί πόντοι είναι -10.

Solution

Απροσδόκητα, η βέλτιστη λύση στο πρόβλημα δεν βασίζεται στις προθεσμίες, αλλά η σωστή άπληστη στρατηγική είναι άπλα να εκτελέσουμε τις διεργασίες βάση της διάρκειας τους σε αύξουσα σειρά Ο λόγος για αυτό είναι αν εκτελέσουμε 2 διεργασίες μια μετά την άλλη και η 1η διεργασία παίρνει περισσότερη χρόνο από την δεύτερη διεργασία, μπορούμε να πάρουμε καλύτερη λύση αν ανταλλάξουμε τις διεργασίες. Για παράδειγμα ας σκεφτούμε το επόμενο χρονοδιάγραμμα:



Εδώ a>b, έτσι πρέπει να ανταλλάξουμε τις διεργασίες:



Solution

Τώρα το X δίνει b λιγότερους πόντους και το Y δίνει a περισσότερους πόντους, έτσι οι συνολικοί πόντοι αυξάνονται κατά $\alpha - \beta > 0$. Σε μια βέλτιστη λύση, για κάθε συνεχόμενες διεργασίες, πρέπει να εκτελείτε η συντομότερη διεργασία πριν την μακρόχρονη διεργασία. Έτσι οι διεργασίες πρέπει να επεξεργαστούν σε αύξουσα σειρά βάση της διάρκειας.

“The best things you would do if you were to die tomorrow are different from the best things you would do if you were to live a full, healthy life.”

adamos2468@gmail.com