Αλγόριθμοι και Πολυπλοκότητα

Άπληστοι Αλγόριθμοι και Δυναμικός Προγραμματισμός Ασκήσεις

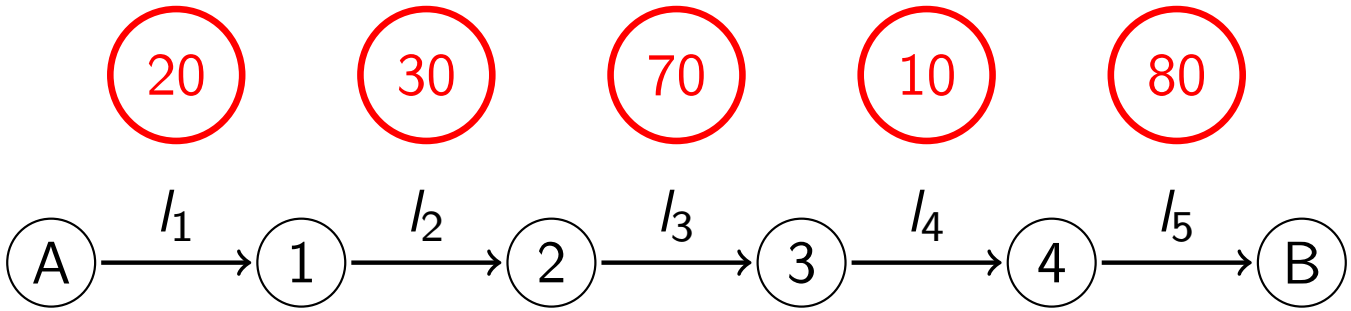
CoReLab

ΣΗΜΜΥ - Ε.Μ.Π.

16 Νοεμβρίου 2016

Outline

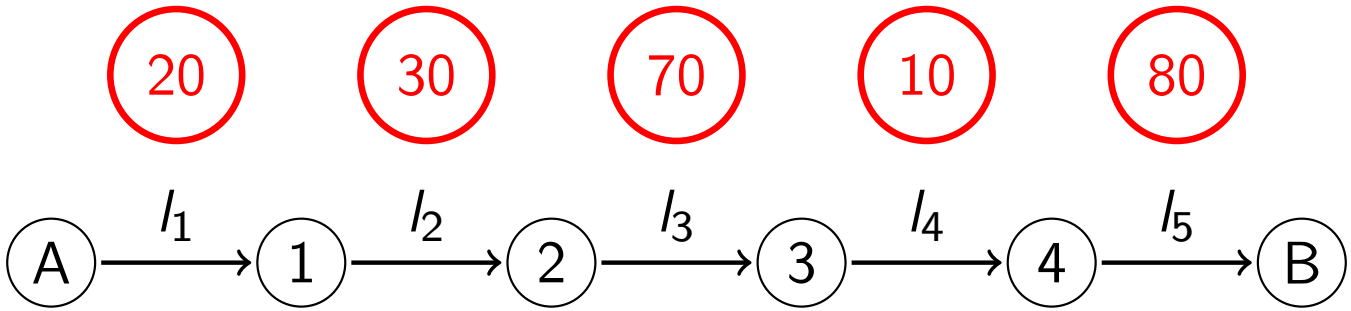
Βιαστικός Μοτοσυκλετιστής



Είσοδος:

- n μήκη διαστημάτων
- n όρια ταχύτητας
- μία ταχύτητα u
- ένα χρονικό όριο T

Βιαστικός Μοτοσυκλετιστής



Είσοδος:

- n μήκη διαστημάτων
- n όρια ταχύτητας
- μία ταχύτητα u
- ένα χρονικό όριο T

Έξοδος: Ο ελάχιστος χρόνος που χρειάζεται για να διανύσει την διαδρομή, αν επιτρέπεται να παραβεί τα όρια ταχύτητας μόνο για συνολική διάρκεια T και κατά ταχύτητα u.

Βιαστικός Μοτοσυκλετιστής

Χρόνος για να διανύσει ένα διάστημα i

Αν ξεπεράσει το όριο κατά u για χρόνο t_i,

$$dt_i = t_i + \frac{l_i - (u_i + u) \cdot t_i}{u_i} = \frac{l_i}{u_i} - \frac{u}{u_i} t_i$$

Βιαστικός Μοτοσυκλετιστής

Χρόνος για να διανύσει ένα διάστημα i

Αν ξεπεράσει το όριο κατά u για χρόνο t_i,

$$dt_i = t_i + \frac{l_i - (u_i + u) \cdot t_i}{u_i} = \frac{l_i}{u_i} - \frac{u}{u_i} t_i$$

Δεδομένου ότι ο μοτοσυκλετιστής δεν επηρεάζει τον όρο $\frac{l_i}{u_i}$, στόχος μας είναι

$$max \sum \frac{u}{u_i} t_i$$

$$s.t. \sum t_i = T$$

$$t_i \leq \frac{l_i}{u_i + u}$$

Βιαστικός Μοτοσυκλετιστής

Βέλτιστη Λύση

Ξοδεύω όσο πιο πολύ χρόνο μπορώ σε διαστήματα με μεγάλο $\frac{u}{u_i}$, δηλαδή μικρό u_i.

Βιαστικός Μοτοσυκλετιστής

Βέλτιστη Λύση

Ξοδεύω όσο πιο πολύ χρόνο μπορώ σε διαστήματα με μεγάλο $\frac{u}{u_i}$, δηλαδή μικρό u_i.

Απόδειξη - Επιχείρημα Ανταλλαγής

Έστω μία βέλτιστη λύση η οποία ορίζει χρόνους t_i^*, και η δικιά μας λύση t_i, όπου τα διαστήματα είναι ταξινομημένα με αύξοντα όρια ταχύτητας.

Βιαστικός Μοτοσυκλετιστής

Βέλτιστη Λύση

Ξοδεύω όσο πιο πολύ χρόνο μπορώ σε διαστήματα με μεγάλο $\frac{u}{u_i}$, δηλαδή μικρό u_i.

Απόδειξη - Επιχείρημα Ανταλλαγής

Έστω μία βέλτιστη λύση η οποία ορίζει χρόνους t_i^*, και η δικιά μας λύση t_i, όπου τα διαστήματα είναι ταξινομημένα με αύξοντα όρια ταχύτητας. Έστω k το διάστημα που διαφέρουν. Σίγουρα $t_k^* < t_k$, καθώς η λύση μας διαθέτει όσο το δυνατόν παραπάνω χρόνο μπορεί σε κάθε διάστημα με τη σειρά.

Βιαστικός Μοτοσυκλετιστής

Βέλτιστη Λύση

Ξοδεύω όσο πιο πολύ χρόνο μπορώ σε διαστήματα με μεγάλο $\frac{u}{u_i}$, δηλαδή μικρό u_i.

Απόδειξη - Επιχείρημα Ανταλλαγής

Έστω μία βέλτιστη λύση η οποία ορίζει χρόνους t_i^*, και η δικιά μας λύση t_i, όπου τα διαστήματα είναι ταξινομημένα με αύξοντα όρια ταχύτητας. Έστω k το διάστημα που διαφέρουν. Σίγουρα $t_k^* < t_k$, καθώς η λύση μας διαθέτει όσο το δυνατόν παραπάνω χρόνο μπορεί σε κάθε διάστημα με τη σειρά.
Αν μεταφέρουμε συνολικό χρόνο $t_k - t_k^*$ από επόμενα διαστήματα στο διάστημα k, ο συνολικός χρόνος δεν χειροτερεύει και η λύση παραμένει βέλτιστη.

Βιαστικός Μοτοσυκλετιστής

Βέλτιστη Λύση

Ξοδεύω όσο πιο πολύ χρόνο μπορώ σε διαστήματα με μεγάλο $\frac{u}{u_i}$, δηλαδή μικρό u_i.

Απόδειξη - Επιχείρημα Ανταλλαγής

Έστω μία βέλτιστη λύση η οποία ορίζει χρόνους t_i^*, και η δικιά μας λύση t_i, όπου τα διαστήματα είναι ταξινομημένα με αύξοντα όρια ταχύτητας. Έστω k το διάστημα που διαφέρουν. Σίγουρα $t_k^* < t_k$, καθώς η λύση μας διαθέτει όσο το δυνατόν παραπάνω χρόνο μπορεί σε κάθε διάστημα με τη σειρά.
Αν μεταφέρουμε συνολικό χρόνο $t_k - t_k^*$ από επόμενα διαστήματα στο διάστημα k, ο συνολικός χρόνος δεν χειροτερεύει και η λύση παραμένει βέλτιστη.
Με την ίδια διαδικασία μπορούμε να μεταασχηματίσουμε τη βέλτιστη λύση στη δική μας χωρίς να αυξήσουμε το συνολικό χρόνο. □

Outline

Επιτροπή Αντιπροσώπων (ΚΤ 4.15)

Είσοδος: n το πλήθος βάρδιες, η κάθε μία εκ των οποίων αντιστοιχεί σε έναν εθελοντή (οι βάρδιες μπορεί να επικαλύπτονται μεταξύ τους).

Έξοδος: Ο μικρότερος αριθμός εθελοντών που μπορεί να σχηματίσει μία πλήρη επιτροπή αντιπροσώπων. Μία επιτροπή είναι πλήρης, αν για κάθε εθελοντή που δεν είναι μέλος της, υπάρχει ένας εθελοντής που είναι μέλος της και η βάρδιά του έχει επικάλυψη με του πρώτου.

Ισοδύναμα

Είσοδος: Διαστήματα $[s_i, f_i)$ $i = 1, \ldots, n$.

Έξοδος: Σύνολο διαστημάτων, ελαχίστου δυνατού μεγέθους, η ένωση των οποίων έχει επικάλυψη με κάθε διάστημα.

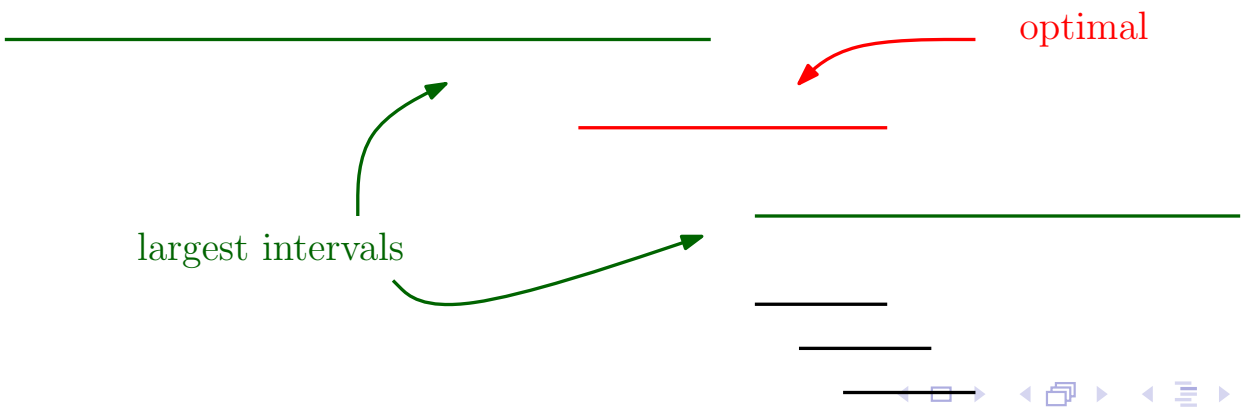
Επιτροπή Αντιπροσώπων

Είσοδος: Διαστήματα $[s_i, f_i)$ $i = 1, \ldots, n$.

Έξοδος: Σύνολο διαστημάτων, ελαχίστου δυνατού μεγέθους, η ένωση των οποίων έχει επικάλυψη με κάθε διάστημα.

Λάθος Ιδέα:
Όσο υπάρχουν ακάλυπτα διαστήματα, διάλεξε το μεγαλύτερο διάστημα που μπορεί να τα καλύψει.

Αντιπαράδειγμα:



Επιτροπή Αντιπροσώπων

Είσοδος: Διαστήματα $[s_i, f_i)$ $i = 1, \ldots, n$.

Έξοδος: Σύνολο διαστημάτων, ελαχίστου δυνατού μεγέθους, η ένωση των οποίων έχει επικάλυψη με κάθε διάστημα.

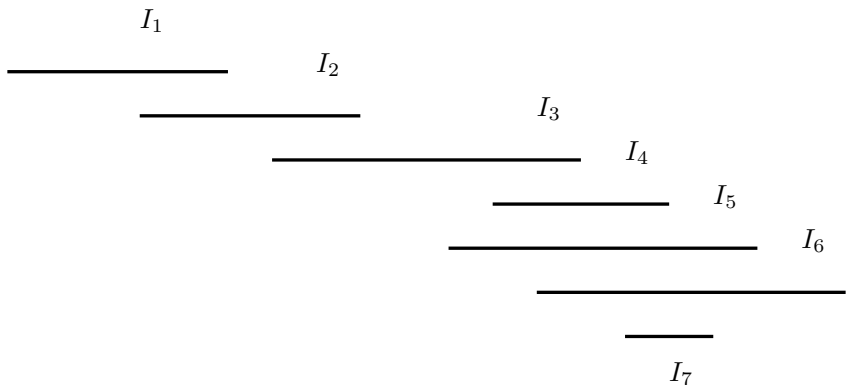
Λύση: Όσο υπάρχουν ακάλυπτα διαστήματα:

- Βρες το ακάλυπτο διάστημα που τελειώνει νωρίτερα.
- Από τα διαστήματα που μπορούν να το καλύψουν, διάλεξε αυτό που τελειώνει αργότερα.

Επιτροπή Αντιπροσώπων

Λύση: Όσο υπάρχουν ακάλυπτα διαστήματα:

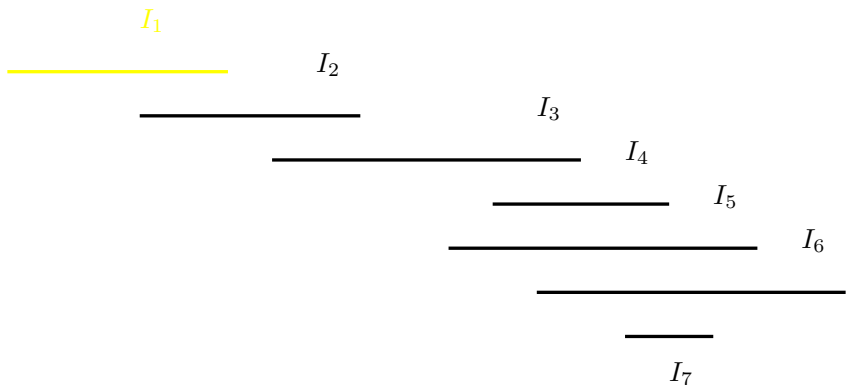
- Βρες το ακάλυπτο διάστημα που τελειώνει νωρίτερα.
- Από τα διαστήματα που μπορούν να το καλύψουν, διάλεξε αυτό που τελειώνει αργότερα.



Επιτροπή Αντιπροσώπων

Λύση: Όσο υπάρχουν ακάλυπτα διαστήματα:

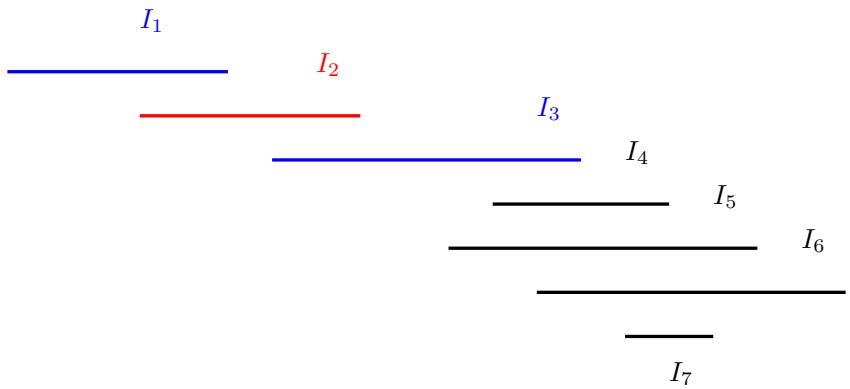
- Βρες το ακάλυπτο διάστημα που τελειώνει νωρίτερα.
- Από τα διαστήματα που μπορούν να το καλύψουν, διάλεξε αυτό που τελειώνει αργότερα.



Επιτροπή Αντιπροσώπων

Λύση: Όσο υπάρχουν ακάλυπτα διαστήματα:

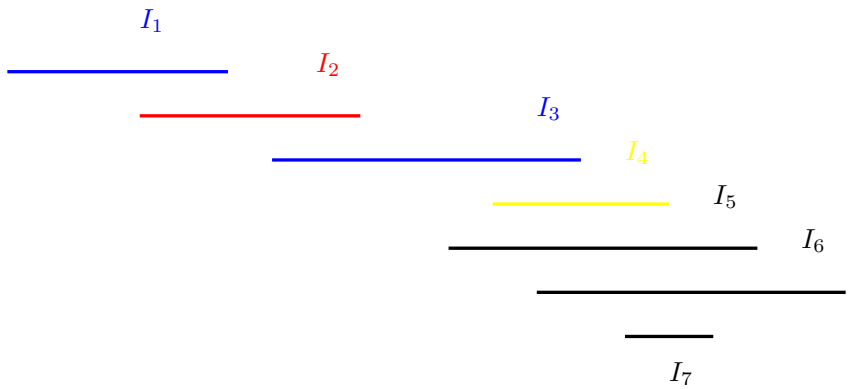
- Βρες το ακάλυπτο διάστημα που τελειώνει νωρίτερα.
- Από τα διαστήματα που μπορούν να το καλύψουν, διάλεξε αυτό που τελειώνει αργότερα.



Επιτροπή Αντιπροσώπων

Λύση: Όσο υπάρχουν ακάλυπτα διαστήματα:

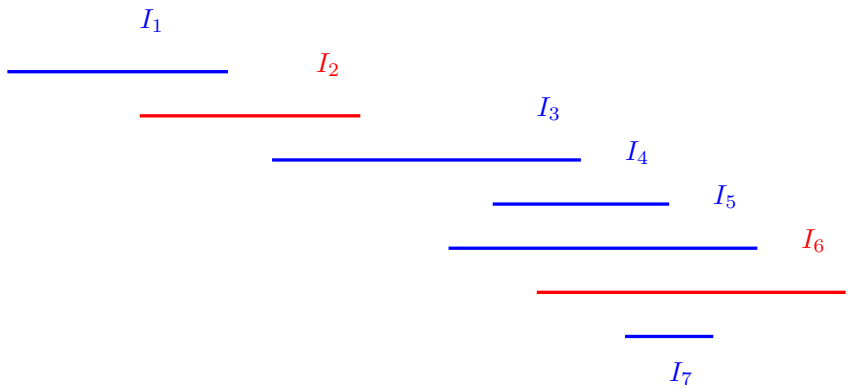
- Βρες το ακάλυπτο διάστημα που τελειώνει νωρίτερα.
- Από τα διαστήματα που μπορούν να το καλύψουν, διάλεξε αυτό που τελειώνει αργότερα.



Επιτροπή Αντιπροσώπων

Λύση: Όσο υπάρχουν ακάλυπτα διαστήματα:

- Βρες το ακάλυπτο διάστημα που τελειώνει νωρίτερα.
- Από τα διαστήματα που μπορούν να το καλύψουν, διάλεξε αυτό που τελειώνει αργότερα.



Επιτροπή Αντιπροσώπων

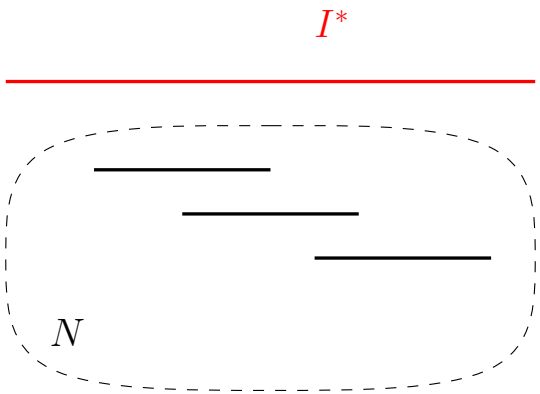
Απόδειξη Ορθότητας - Επιχείρημα Ανταλλαγής

- Έστω μία βέλτιστη λύση O^* και O η δική μας. Έστω ότι ταξινομούμε σε αύξουσα σειρά τα διαστήματά μας σύμφωνα με το χρόνο ολοκλήρωσής τους και το ίδιο κάνουμε και για τα διαστήματα της βέλτιστης λύσης. Έστω I^* το πρώτο διάστημα στο οποίο οι δύο λύσεις διαφέρουν.

Επιτροπή Αντιπροσώπων

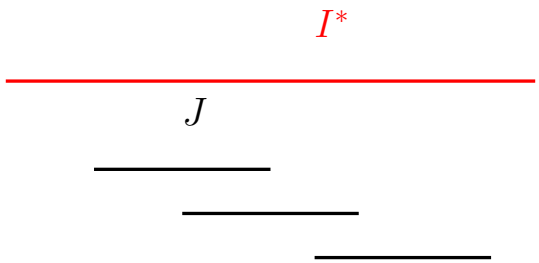
Απόδειξη Ορθότητας - Επιχείρημα Ανταλλαγής

- Έστω N το σύνολο των διαστημάτων που καλύπτονται μόνο από το I^* στη βέλτιστη λύση. Προφανώς, $N \neq \emptyset$, αφού διαφορετικά δε θα ήταν βέλτιστη γιατί θα μπορούσαμε να αποκλείσουμε το I^* από τη λύση και να παίρναμε εφικτή λύση μικρότερου μεγέθους.



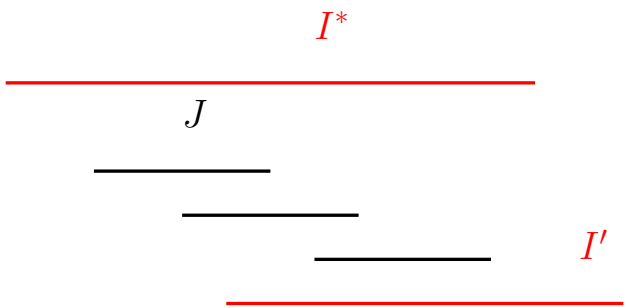
Απόδειξη Ορθότητας - Επιχείρημα Ανταλλαγής

- Έστω J το διάστημα του N που τελειώνει νωρίτερα.



Απόδειξη Ορθότητας - Επιχείρημα Ανταλλαγής

- Το J καλύπτεται μόνο από το I^* στη βέλτιστη λύση άρα το προηγούμενο διάστημα αυτής δεν το καλύπτει. Επομένως, αφού το I^* είναι το πρώτο διάστημα στο οποίο οι δύο λύσεις διαφέρουν, το J θα είναι μέχρι στιγμής ακάλυπτο και στη δική μας λύση. Ο δικός μας αλγόριθμος, αφού παράγει εφικτή λύση, για να καλύψει το ακάλυπτο J θα έχει επιλέξει ένα διάστημα I' που έχει επικάλυψη με το J και, λόγω του κριτηρίου επιλογής, θα τελειώνει τουλάχιστον το ίδιο αργά με το I^*. Άρα θα καλύπτει τουλάχιστον όλο το N.



Επιτροπή Αντιπροσώπων

Απόδειξη Ορθότητας - Επιχείρημα Ανταλλαγής

Ανταλλάσσοντας το I^* με το I' επομένως θα είχαμε μία εφικτή λύση O' ίδιου μεγέθους με τη βέλτιστη, δηλαδή θα ήταν κι αυτή βέλτιστη λύση. Συνεχίζοντας την ίδια διαδικασία, πλησιάζουμε όλο και πιο πολύ τη δική μας λύση, κρατώντας το μέγεθος ελάχιστο. Άρα και η δική μας λύση είναι βέλτιστη αφού περιλαμβάνει όλα τα διαστήματα της βέλτιστης εκτός από κάποια που αντικαθίστανται (1 προς 1) από τα I', I'', κ.ο.κ. και κανένα άλλο διάστημα*.

* Έστω ότι καθώς διατρέχουμε τις δύο λίστες με τα διαστήματα των δύο λύσεων, φθάνουμε στο τέλος της βέλτιστης και η δική μας έχει ένα ακόμα διάστημα E. Το E τελειώνει αργότερα από το τελευταίο μας μέχρι στιγμής. Το E όμως δε μπορεί να επιλέχθηκε για να καλύψει διάστημα το οποίο αρχίζει πριν το τέλος του τελευταίου μας, γιατί έχοντας το μεγαλύτερο χρόνο ολοκλήρωσης απ'όλα θα είχε επιλεγεί στη θέση κάποιων προηγούμενων και άρα θα ήταν στη διαδικασία ανταλλαγής. Επίσης, δε μπορεί να επιλέχθηκε για να καλύψει διάστημα που αρχίζει μετά το τέλος του τελευταίου μας, αφού μέχρι τότε έχουμε καλύψει τουλάχιστον τόσα διαστήματα όσα καλύπτει η βέλτιστη λύση, δηλαδή όλα. Άρα δεν υπάρχει τέτοιο διάστημα.

Επιτροπή Αντιπροσώπων

Χρονική Πολυπλοκότητα

- Φτιάχνουμε δύο πίνακες S, F με τα διαστήματα ταξινομημένα ως προς τους χρόνους έναρξης και λήξης αντίστοιχα, σε χρόνο $O(n \log n)$.
- Χρησιμοποιούμε τον πρώτο για να βρίσκουμε τα ακάλυπτα διαστήματα και τον δεύτερο για να επιλέξουμε ποιό θα βάλουμε στη λύση μας.
- Στους πίνακες αυτούς αντιστοιχούν δείκτες p_s, p_f οι οποίοι, όπως θα φανεί, κινούνται πάντα προς τα δεξιά άρα η διαδικασία είναι γραμμικού χρόνου ως προς το n.

Συνολικά, ο αλγόριθμός μας έχει πολυπλοκότητα:

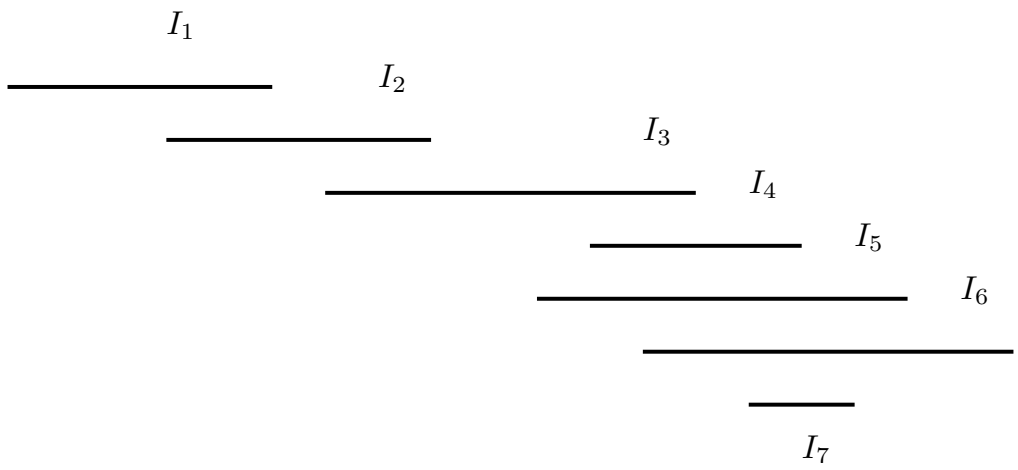
$$O(n \log n)$$

Επιτροπή Αντιπροσώπων

Χρονική Πολυπλοκότητα

$$F = \begin{bmatrix} f_1 & f_2 & f_3 & f_4 & f_7 & f_5 & f_6 \end{bmatrix} \qquad p_f = 1$$

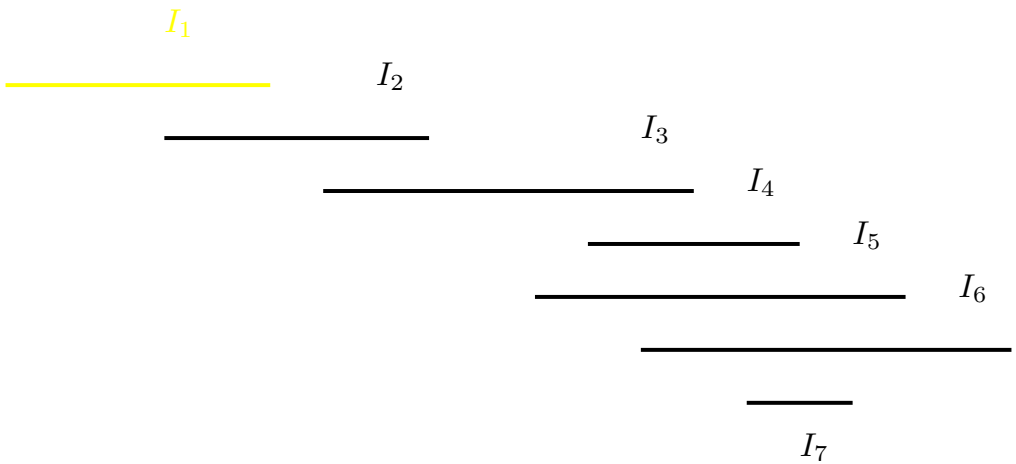
$$S = \begin{bmatrix} s_1 & s_2 & s_3 & s_5 & s_4 & s_6 & s_7 \end{bmatrix} \qquad p_s = 1$$



Επιτροπή Αντιπροσώπων

Χρονική Πολυπλοκότητα

$$F = \begin{bmatrix} f_1 & f_2 & f_3 & f_4 & f_7 & f_5 & f_6 \end{bmatrix} \qquad p_f = 1$$
$$S = \begin{bmatrix} s_1 & s_2 & s_3 & s_5 & s_4 & s_6 & s_7 \end{bmatrix} \qquad p_s = 1$$



Επιτροπή Αντιπροσώπων

Χρονική Πολυπλοκότητα

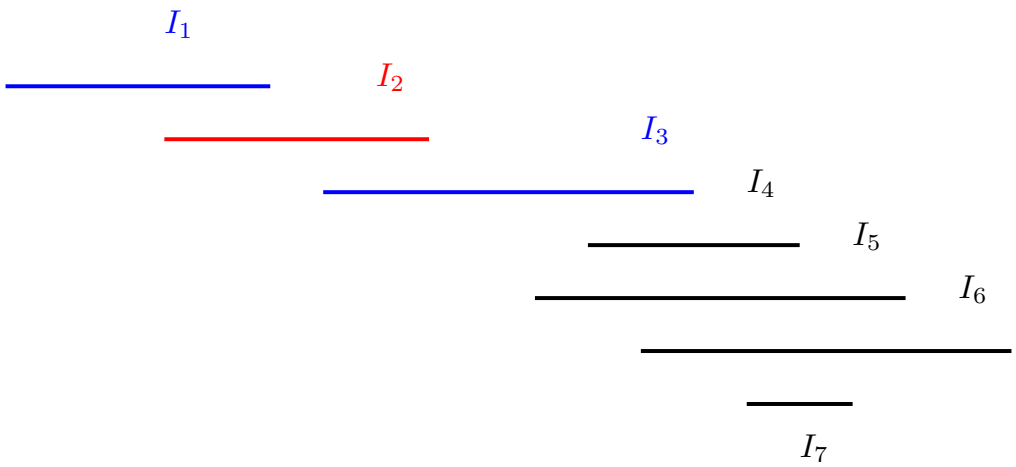
- Κινούμαστε στον S μέχρι να βρούμε όλα τα διαστήματα που έχουν επικάλυψη με το I_1 (δηλαδή μέχρι $s_i \geq f_1$) και παράλληλα αποθηκεύουμε αυτό με το μεγαλύτερο $f_i (= f_2)$. Αυτά τα διαστήματα δεν θα τα ξανακοιτάξουμε αφού τελειώνουν πριν το διάστημα που επιλέξαμε και δεν υπάρχουν ακάλυπτα διαστήματα στα αριστερά τους. Άρα ο δείκτης p_s την επόμενη φορά θα ξεκινήσει από το σημείο που έμεινε.
- Έπειτα, κινούμαστε στον F μέχρι να βρούμε το επόμενο ακάλυπτο διάστημα (μέχρι $s_i \geq f_2$).

Επιτροπή Αντιπροσώπων

Χρονική Πολυπλοκότητα

$$F = \begin{bmatrix} f_1 & f_2 & f_3 & f_4 & f_7 & f_5 & f_6 \end{bmatrix} \qquad p_f = 4$$

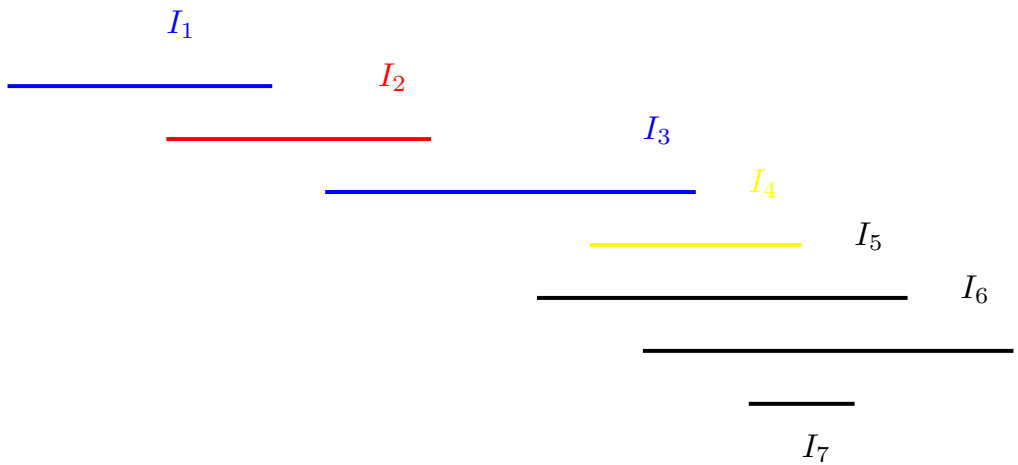
$$S = \begin{bmatrix} s_1 & s_2 & s_3 & s_5 & s_4 & s_6 & s_7 \end{bmatrix} \qquad p_s = 3$$



Επιτροπή Αντιπροσώπων

Χρονική Πολυπλοκότητα

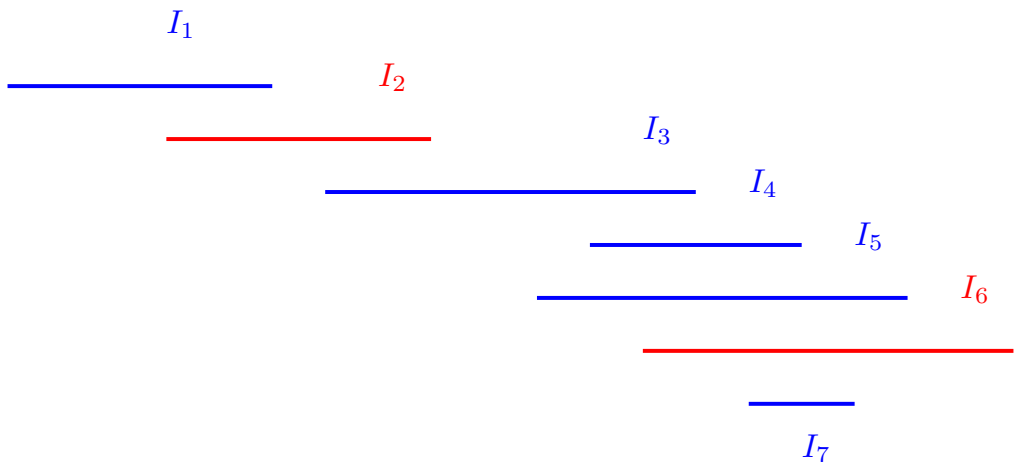
$$F = \begin{bmatrix} f_1 & f_2 & f_3 & f_4 & f_7 & f_5 & f_6 \end{bmatrix} \qquad p_f = 4$$
$$S = \begin{bmatrix} s_1 & s_2 & s_3 & s_5 & s_4 & s_6 & s_7 \end{bmatrix} \qquad p_s = 3$$



Επιτροπή Αντιπροσώπων

Χρονική Πολυπλοκότητα

$$F = \begin{bmatrix} f_1 & f_2 & f_3 & f_4 & f_7 & f_5 & f_6 \end{bmatrix} \qquad p_f = 8$$
$$S = \begin{bmatrix} s_1 & s_2 & s_3 & s_5 & s_4 & s_6 & s_7 \end{bmatrix} \qquad p_s = 8$$



Outline

Βότσαλα στη Σκακιέρα (DPV 6.5)

Είσοδος: Μία σκακιέρα διαστάσεων $4 \times n$ και ένας ακέραιος σε κάθε κουτάκι.

17	202	104	42	54
5	92	71	5	203
200	48	39	0	98
1	29	2	7	6

Βότσαλα στη Σκακιέρα (DPV 6.5)

Είσοδος: Μία σκακιέρα διαστάσεων $4 \times n$ και ένας ακέραιος σε κάθε κουτάκι.

17	202	104	42	54
5	92	71	5	203
200	48	39	0	98
1	29	2	7	6

Στόχος: Να τοποθετήσουμε βότσαλα στα κουτάκια, ώστε κανένα βότσαλο να μην συνορεύει με άλλο οριζόντια ή κάθετα, μεγιστοποιώντας το άθροισμα των αριθμών στα κουτάκια με βότσαλα.

Βότσαλα στη Σκακιέρα (DPV 6.5)

Πρώτη Προσέγγιση

Όσο υπάρχει χώρος, τοποθετούμε ένα βότσαλο στο κουτάκι με τον μεγαλύτερο αριθμό που επιτρέπεται.

Βότσαλα στη Σκακιέρα (DPV 6.5)

Πρώτη Προσέγγιση

Όσο υπάρχει χώρος, τοποθετούμε ένα βότσαλο στο κουτάκι με τον μεγαλύτερο αριθμό που επιτρέπεται.

Αντιπαράδειγμα

0	99	0
99	100	99
0	99	0

Βότσαλα στη Σκακιέρα (DPV 6.5)

Πρώτη Προσέγγιση

Όσο υπάρχει χώρος, τοποθετούμε ένα βότσαλο στο κουτάκι με τον μεγαλύτερο αριθμό που επιτρέπεται.

Αντιπαράδειγμα

0	99	0
99	100	99
0	99	0

αξία greedy 100

Βότσαλα στη Σκακιέρα (DPV 6.5)

Πρώτη Προσέγγιση

Όσο υπάρχει χώρος, τοποθετούμε ένα βότσαλο στο κουτάκι με τον μεγαλύτερο αριθμό που επιτρέπεται.

Αντιπαράδειγμα

0	99	0
99	100	99
0	99	0

βέλτιστη αξία 396

Βότσαλα στη Σκακιέρα (DPV 6.5)

Είναι όμως τόσο τραγικός ο greedy αλγόριθμος;

Βότσαλα στη Σκακιέρα (DPV 6.5)

Είναι όμως τόσο τραγικός ο greedy αλγόριθμος;

Πρόταση

Ο άπληστος αλγόριθμος είναι $\frac{1}{4}$-προσεγγιστικός. Παράγει δηλαδή λύση αξίας τουλάχιστον το 25% της βέλτιστης.

Βότσαλα στη Σκακιέρα (DPV 6.5)

Είναι όμως τόσο τραγικός ο greedy αλγόριθμος;

Πρόταση

Ο άπληστος αλγόριθμος είναι $\frac{1}{4}$-προσεγγιστικός. Παράγει δηλαδή λύση αξίας τουλάχιστον το 25% της βέλτιστης.

Απόδειξη.

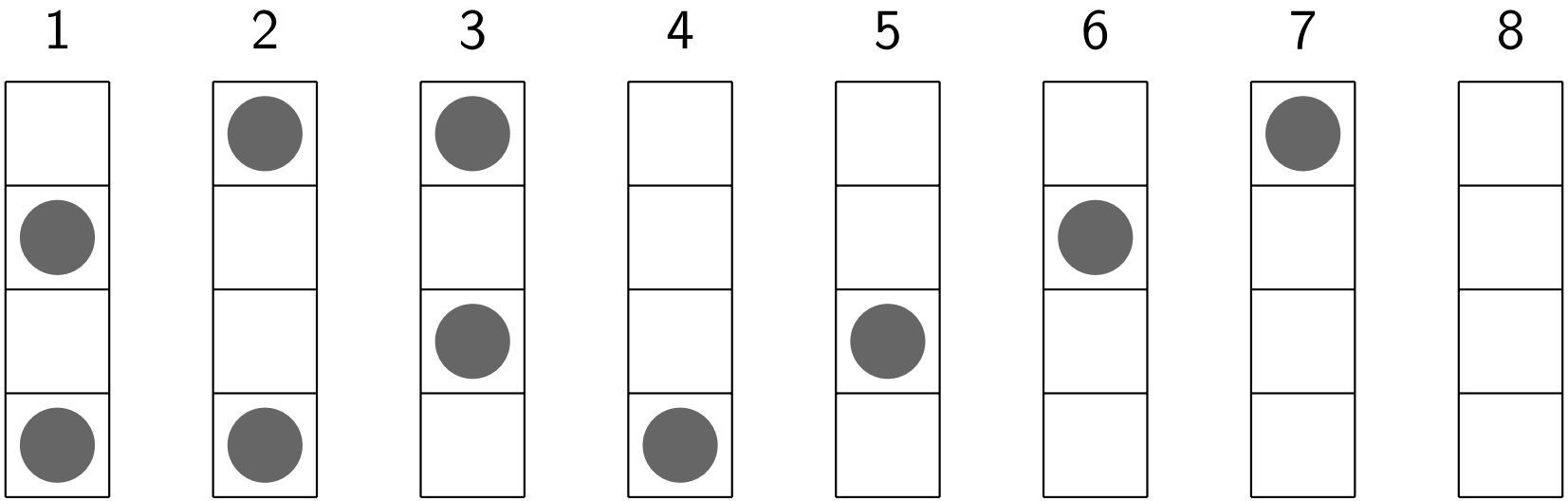
Έστω S^* το σύνολο των βοτσάλων της βέλτιστης λύσης αξίας OPT και S το σύνολο της άπληστης λύσης αξίας GR

$$\begin{aligned} OPT &= \sum(S^* \setminus S) + \sum(S \cap S^*) \leq \\ &\leq 4\sum(S \setminus S^*) + \sum(S \cap S^*) \leq \\ &\leq 4\left(\sum(S \setminus S^*) + \sum(S \cap S^*)\right) = 4GR \end{aligned}$$

□

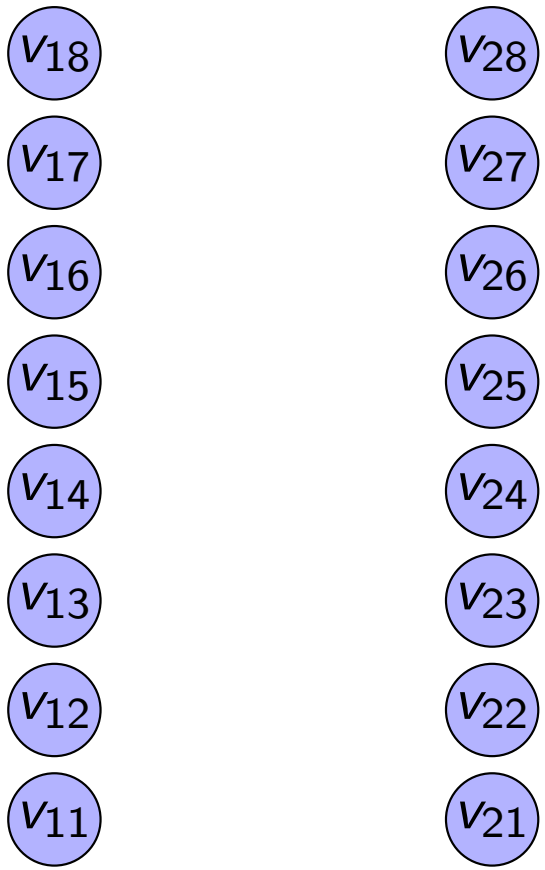
Βότσαλα στη Σκακιέρα (DPV 6.5)

Αφού η σκακιέρα έχει σταθερό αριθμός γραμμών, κάθε στήλη μπορεί να γεμίσει με συγκεκριμένο αριθμό τρόπων.

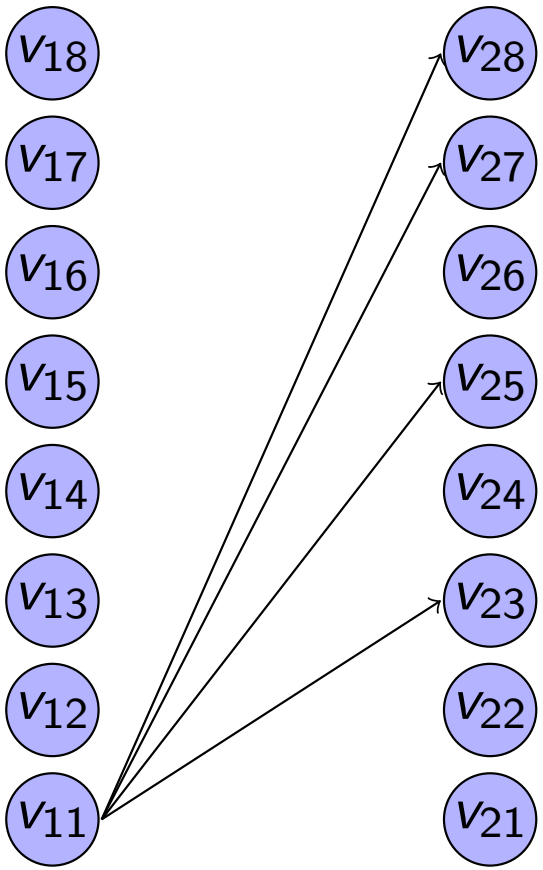


Ο συνδυασμός που θα επιλέξω επηρεάζει τις υπόλοιπες επιλογές.

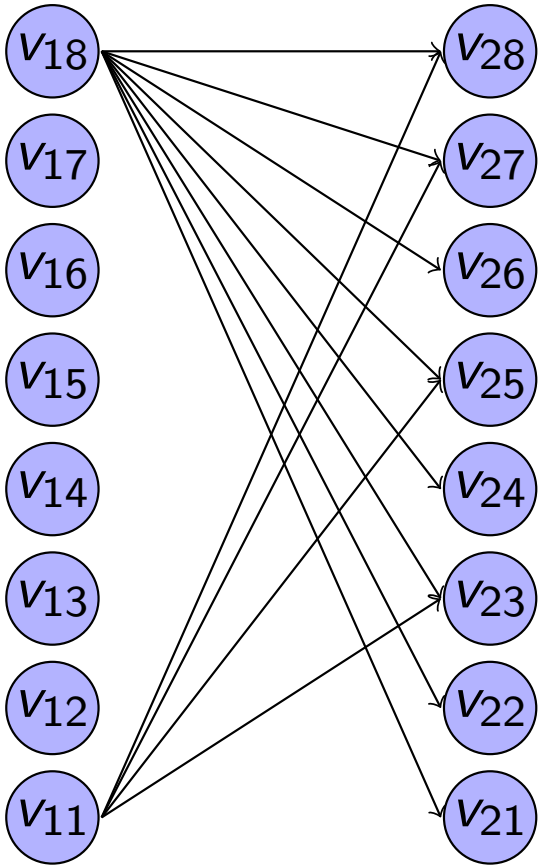
Βότσαλα στη Σκακιέρα (DPV 6.5)



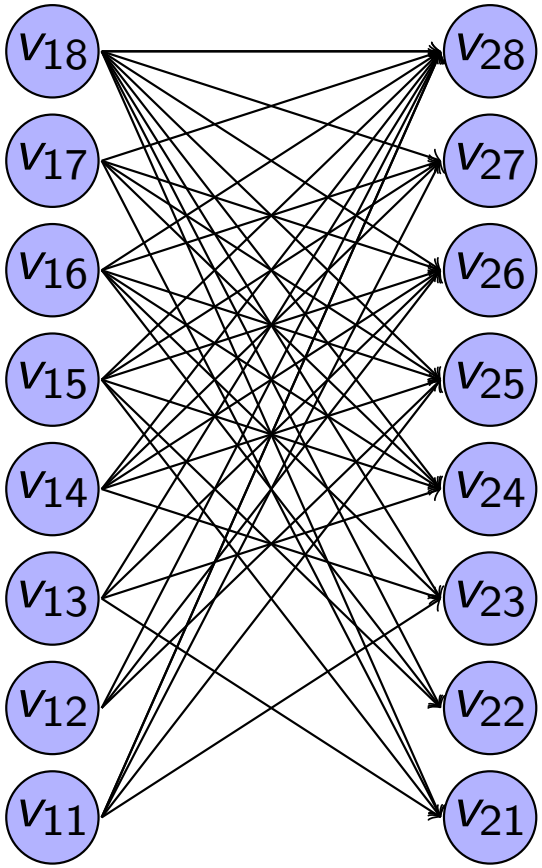
Βότσαλα στη Σκακιέρα (DPV 6.5)



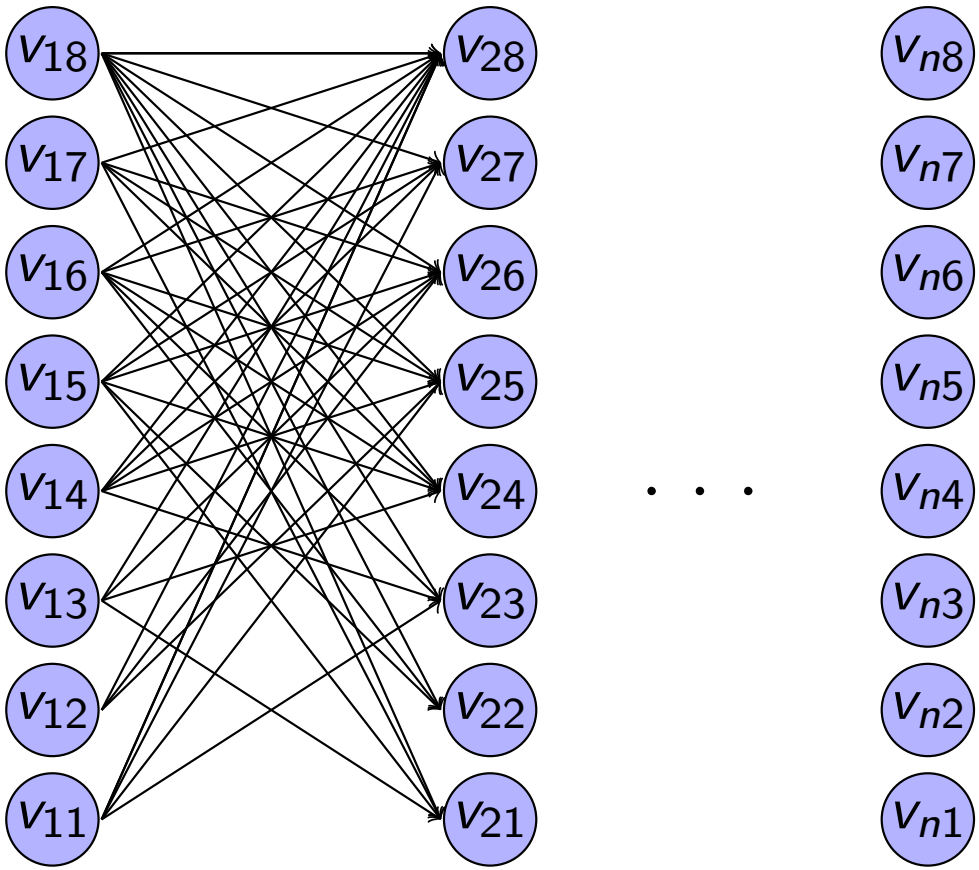
Βότσαλα στη Σκακιέρα (DPV 6.5)



Βότσαλα στη Σκακιέρα (DPV 6.5)



Βότσαλα στη Σκακιέρα (DPV 6.5)



Βότσαλα στη Σκακιέρα (DPV 6.5)

Ιδέα

Για κάθε στήλη, θα λύνω το πρόβλημα βέλτιστα για όλους του πιθανούς συνδυασμούς.

Βότσαλα στη Σκακιέρα (DPV 6.5)

Ιδέα

Για κάθε στήλη, θα λύνω το πρόβλημα βέλτιστα για όλους του πιθανούς συνδυασμούς.

Θα χρησιμοποιήσω δυναμικό προγραμματισμό στη σχέση

$$C[i,j] = v_{ij} + \max_{k \in adj(j)} \{C[i-1,k]\}$$

$$C[1,j] = v_{1j}$$

Βότσαλα στη Σκακιέρα (DPV 6.5)

Ιδέα

Για κάθε στήλη, θα λύνω το πρόβλημα βέλτιστα για όλους του πιθανούς συνδυασμούς.

Θα χρησιμοποιήσω δυναμικό προγραμματισμό στη σχέση

$$C[i,j] = v_{ij} + \max_{k \in adj(j)} \{C[i-1,k]\}$$

$$C[1,j] = v_{1j}$$

Όπου

- $C[i,j]$: η αξία της βέλτιστης λύσης για τις στήλες 1 εώς i όπου η στήλη i έχει συνδυασμό βοτσάλων j
- v_{ij}: η αξία που παράγεται αν τοποθετηθούν βότσαλα σε συνδυασμό j στην στήλη i
- $adj(j)$: οι συνδυασμού που μπορούν να συνορεύουν με το συνδυασμό j

Βότσαλα στη Σκακιέρα (DPV 6.5)

Τελική Λύση

Η τιμή της βέλτιστης λύσης είναι η μέγιστη τιμή της στήλης n

$$OPT = \max_i C[n, i]$$

Η λύση προκύτει από τον ίδιο τον πίνακα γυρνώντας προς τα πίσω από το τέλος.

Χρονική Πολυπλοκότητα

Ο πίνακας C έχει μέγεθος $8 \cdot n$ και αφού ο αριθμός των γραμμών είναι σταθερός, κάθε στήλη υπολογίζεται σε σταθερό χρόνο. Συνολικά

$$\Theta(n)$$

Outline

Μέγιστη Κοινή Υπακολουθία (CLRS ch15.4)

Είσοδος: Δύο ακολουθείες χαρακτήρων

Έξοδος: Η μέγιστη σε μήκος κοινή υποακολουθία

Μέγιστη Κοινή Υπακολουθία (CLRS ch15.4)

Είσοδος: Δύο ακολουθείες χαρακτήρων

Έξοδος: Η μέγιστη σε μήκος κοινή υποακολουθία

B A N A N A

B A T M A N

Μέγιστη Κοινή Υπακολουθία (CLRS ch15.4)

Είσοδος: Δύο ακολουθείες χαρακτήρων

Έξοδος: Η μέγιστη σε μήκος κοινή υποακολουθία

B **A** N **A** **N** A

B **A** T M **A** **N**

Μέγιστη Κοινή Υπακολουθία (CLRS ch15.4)

Πρώτη Προσέγγιση

Κοιτάζω τον πρώτο χαρακτήρα κάθε ακολουθίας:

- Αν είναι ίδιοι, τότε σίγουρα αποτελούν μέρος τη μέγιστης υποακολουθίας
- Αν όχι, πρέπει να πετάξω τον έναν από τους δύο και να συνεχίσω

Πως ξέρω όμως ποιον χαρακτήρα θα αφήσω προκειμένου να έχω βέλτιστη λύση;

Μέγιστη Κοινή Υπακολουθία (CLRS ch15.4)

Έστω $C[i,j]$ το μήκος της μέγιστης κοινής υποακολουθίας που προκύπτει μέχρι και τον χαρακτήρα i της πρώτης ακολουθίας και τον χαρακτήρα j της δεύτερης. Γενικεύοντας της προηγούμενη ιδέα προκύπτει η αναδρομική σχέση

$$C[i,j] = \begin{cases} C[i-1,j-1]+1 & , \text{αν } S_i = S'_j \\ \max(C[i-1,j], C[i,j-1]) & , \text{διαφορετικά} \end{cases}$$

$$C[i,0] = 0$$

$$C[0,j] = 0$$

Όπου S, S' οι δύο ακολουθίες

Μέγιστη Κοινή Υπακολουθία (CLRS ch15.4)

Χρησιμοποιώντας δυναμικό προγραμματισμό, η παραπάνω σχέση υπολογίζεται αποδοτικά

	ε	B	A	N	A	N	A
ε							
B							
A							
T							
M							
A							
N							

Μέγιστη Κοινή Υπακολουθία (CLRS ch15.4)

Χρησιμοποιώντας δυναμικό προγραμματισμό, η παραπάνω σχέση υπολογίζεται αποδοτικά

	ε	B	A	N	A	N	A
ε	0	0	0	0	0	0	0
B	0						
A	0						
T	0						
M	0						
A	0						
N	0						

Μέγιστη Κοινή Υπακολουθία (CLRS ch15.4)

Χρησιμοποιώντας δυναμικό προγραμματισμό, η παραπάνω σχέση υπολογίζεται αποδοτικά

	ε	B	A	N	A	N	A
ε	0	0	0	0	0	0	0
B	0	1					
A	0						
T	0						
M	0						
A	0						
N	0						

Μέγιστη Κοινή Υπακολουθία (CLRS ch15.4)

Χρησιμοποιώντας δυναμικό προγραμματισμό, η παραπάνω σχέση υπολογίζεται αποδοτικά

	ε	B	A	N	A	N	A
ε	0	0	0	0	0	0	0
B	0	1	1	1	1	1	1
A	0						
T	0						
M	0						
A	0						
N	0						

Μέγιστη Κοινή Υπακολουθία (CLRS ch15.4)

Χρησιμοποιώντας δυναμικό προγραμματισμό, η παραπάνω σχέση υπολογίζεται αποδοτικά

	ε	B	A	N	A	N	A
ε	0	0	0	0	0	0	0
B	0	1	1	1	1	1	1
A	0	1					
T	0						
M	0						
A	0						
N	0						

Μέγιστη Κοινή Υπακολουθία (CLRS ch15.4)

Χρησιμοποιώντας δυναμικό προγραμματισμό, η παραπάνω σχέση υπολογίζεται αποδοτικά

	ε	B	A	N	A	N	A
ε	0	0	0	0	0	0	0
B	0	1	1	1	1	1	1
A	0	1	2				
T	0						
M	0						
A	0						
N	0						

Μέγιστη Κοινή Υπακολουθία (CLRS ch15.4)

Χρησιμοποιώντας δυναμικό προγραμματισμό, η παραπάνω σχέση υπολογίζεται αποδοτικά

	ε	B	A	N	A	N	A
ε	0	0	0	0	0	0	0
B	0	1	1	1	1	1	1
A	0	1	2	2	2	2	2
T	0						
M	0						
A	0						
N	0						

Μέγιστη Κοινή Υπακολουθία (CLRS ch15.4)

Χρησιμοποιώντας δυναμικό προγραμματισμό, η παραπάνω σχέση υπολογίζεται αποδοτικά

	ε	B	A	N	A	N	A
ε	0	0	0	0	0	0	0
B	0	1	1	1	1	1	1
A	0	1	2	2	2	2	2
T	0	1	2	2	2	2	2
M	0						
A	0						
N	0						

Μέγιστη Κοινή Υπακολουθία (CLRS ch15.4)

Χρησιμοποιώντας δυναμικό προγραμματισμό, η παραπάνω σχέση υπολογίζεται αποδοτικά

	ε	B	A	N	A	N	A
ε	0	0	0	0	0	0	0
B	0	1	1	1	1	1	1
A	0	1	2	2	2	2	2
T	0	1	2	2	2	2	2
M	0	1	2	2	2	2	2
A	0						
N	0						

Μέγιστη Κοινή Υπακολουθία (CLRS ch15.4)

Χρησιμοποιώντας δυναμικό προγραμματισμό, η παραπάνω σχέση υπολογίζεται αποδοτικά

	ε	B	A	N	A	N	A
ε	0	0	0	0	0	0	0
B	0	1	1	1	1	1	1
A	0	1	2	2	2	2	2
T	0	1	2	2	2	2	2
M	0	1	2	2	2	2	2
A	0	1	2	2	3	3	3
N	0						

Μέγιστη Κοινή Υπακολουθία (CLRS ch15.4)

Χρησιμοποιώντας δυναμικό προγραμματισμό, η παραπάνω σχέση υπολογίζεται αποδοτικά

	ε	B	A	N	A	N	A
ε	0	0	0	0	0	0	0
B	0	1	1	1	1	1	1
A	0	1	2	2	2	2	2
T	0	1	2	2	2	2	2
M	0	1	2	2	2	2	2
A	0	1	2	2	3	3	3
N	0	1	2	3	3	4	4

Μέγιστη Κοινή Υπακολουθία (CLRS ch15.4)

Χρησιμοποιώντας δυναμικό προγραμματισμό, η παραπάνω σχέση υπολογίζεται αποδοτικά

	ε	B	A	N	A	N	A
ε	0	0	0	0	0	0	0
B	0	1	1	1	1	1	1
A	0	1	2	2	2	2	2
T	0	1	2	2	2	2	2
M	0	1	2	2	2	2	2
A	0	1	2	2	3	3	3
N	0	1	2	3	3	4	4

Μέγιστη Κοινή Υπακολουθία (CLRS ch15.4)

Τελική Λύση

Η τιμή της βέλτιστης λύσης βρίσκεται στο $C[n, m]$, ενώ από τον πίνακα μπορεί να βρεθεί πως προέκυψε κάθε τιμή. Τα στοιχεία της κοινής υποακολουθίας είναι ακριβώς αυτά που έγινα 'διαγώνιες' μεταβάσεις.

Χρονική Πολυπλοκότητα

Ο πίνακας είναι διαστάσεων $n \cdot m$ και κάθε κελί γεμίζει σε $O(1)$, άρα συνολικά

$$\Theta(n \cdot m)$$

Outline

Μέγιστη Αύξουσα Υπακολουθία (DPV ch6.2)

Είσοδος: Ακολουθία n αριθμών: $a_1, \ldots, a_n$.

Μέγιστη Αύξουσα Υπακολουθία (DPV ch6.2)

Είσοδος: Ακολουθία n αριθμών: $a_1, \ldots, a_n$.

Έξοδος: Μέγιστη (γνησίως) αύξουσα υπακολουθία.

Μέγιστη Αύξουσα Υπακολουθία (DPV ch6.2)

Είσοδος: Ακολουθία n αριθμών: $a_1, \ldots, a_n$.

Έξοδος: Μέγιστη (γνησίως) αύξουσα υπακολουθία.

π.χ.
Ακολουθία ($n = 6$): 5-2-6-3-6-9, δηλαδή
$a_1 = 5, a_2 = 2, a_3 = 6, a_4 = 3, a_5 = 6, a_6 = 9$
Μέγιστη αύξουσα υπακολουθία: 2-3-6-9

Μέγιστη Αύξουσα Υπακολουθία (DPV ch6.2)

Είσοδος: Ακολουθία n αριθμών: $a_1, \ldots, a_n$.

Έξοδος: Μέγιστη (γνησίως) αύξουσα υπακολουθία.

Λύση με χρονική πολυπλοκότητα $O(n^2)$:

Από όλες τις αύξουσες υπακολουθίες που τελειώνουν στη θέση $j \in \{1, \ldots, n\}$, διαλέγω τη μεγαλύτερη, μήκους έστω $L(j)$. Έπειτα, διαλέγω από τις τελευταίες και πάλι τη μεγαλύτερη, δηλαδή το κατάλληλο i τέτοιο ώστε $i = \text{argmax}_{j \in \{1,\ldots,n\}} L(j)$.

Μέγιστη Αύξουσα Υπακολουθία (DPV ch6.2)

Λύση με χρονική πολυπλοκότητα $O(n^2)$:

Αρκεί να βρούμε τον αναδρομικό τύπο για το $L(j)$.

Μέγιστη Αύξουσα Υπακολουθία (DPV ch6.2)

Λύση με χρονική πολυπλοκότητα $O(n^2)$:

Αρκεί να βρούμε τον αναδρομικό τύπο για το $L(j)$.

- Φτιάχνουμε λίστες list(i) $i = 1, \ldots, n$ με τα στοιχεία που ικανοποιούν $a_j < a_i$ και $j < i$ σε χρόνο $O(n^2)$ (κάθε αύξουσα υπακολουθία που τελειώνει στο i θα έχει προτελευταίο στοιχείο κάποιο από τη list(i)).

Μέγιστη Αύξουσα Υπακολουθία (DPV ch6.2)

Λύση με χρονική πολυπλοκότητα $O(n^2)$:

Αρκεί να βρούμε τον αναδρομικό τύπο για το $L(j)$.

- Φτιάχνουμε λίστες list(i) $i = 1, \ldots, n$ με τα στοιχεία που ικανοποιούν $a_j < a_i$ και $j < i$ σε χρόνο $O(n^2)$ (κάθε αύξουσα υπακολουθία που τελειώνει στο i θα έχει προτελευταίο στοιχείο κάποιο από τη list(i)).
- Η μέγιστη αύξουσα υπακολουθία που τελειώνει στο j θα έχει μήκος ένα μεγαλύτερο από τη μέγιστη υπακολουθία που τελειώνει σε κάποιο στοιχείο του list(i).

Μέγιστη Αύξουσα Υπακολουθία (DPV ch6.2)

Λύση με χρονική πολυπλοκότητα $O(n^2)$:

Άρα

$$L(j) = 1 + \max\{L(i) \mid i \in \text{list}(j)\}$$

Μέγιστη Αύξουσα Υπακολουθία (DPV ch6.2)

Λύση με χρονική πολυπλοκότητα $O(n^2)$:

Άρα

$$L(j) = 1 + \max\{L(i) \mid i \in \text{list}(j)\}$$

- Για τον υπολογισμό κάθε ενός από τα $L(i)$ συνεπώς χρειάζεται χρόνος $|\text{list}(i)| \leq n$ άρα συνολικά ο αλγόριθμός μας έχει πολυπλοκότητα $O(n^2)$.

Μέγιστη Αύξουσα Υπακολουθία (DPV ch6.2)

Λύση με χρονική πολυπλοκότητα $O(n^2)$:

Άρα

$$L(j) = 1 + \max\{L(i) \mid i \in \text{list}(j)\}$$

- Για τον υπολογισμό κάθε ενός από τα $L(i)$ συνεπώς χρειάζεται χρόνος $|\text{list}(i)| \leq n$ άρα συνολικά ο αλγόριθμός μας έχει πολυπλοκότητα $O(n^2)$.
- Για να βρούμε την υπακολουθία αρκεί να αποθηκεύουμε ως προηγούμενο στοιχείο του j το στοιχείο που επιλέγεται στον υπολογισμό του $L(j)$.

Μέγιστη Αύξουσα Υπακολουθία (DPV ch6.2)

Άλλη λύση με χρονική πολυπλοκότητα $O(n^2)$:

Μπορούμε να χρησιμοποιήσουμε κάπως το πρόβλημα της Μέγιστης Κοινής Υπακολουθίας;

Μέγιστη Αύξουσα Υπακολουθία (DPV ch6.2)

Άλλη λύση με χρονική πολυπλοκότητα $O(n^2)$:

Μπορούμε να χρησιμοποιήσουμε κάπως το πρόβλημα της Μέγιστης Κοινής Υπακολουθίας;

- Φτιάχνουμε την ακολουθία $a'_1, \ldots, a'_m$ η οποία αποτελείται από τα διαφορετικά στοιχεία της αρχικής ακολουθίας ταξινομημένα κατά αύξουσα σειρά.
- Η Μέγιστη Κοινή Υπακολουθία μεταξύ της ταξινομημένης και της αρχικής είναι και η ζητούμενη Μέγιστη Αύξουσα Υπακολουθία.

Μέγιστη Αύξουσα Υπακολουθία (DPV ch6.2)

Άλλη λύση με χρονική πολυπλοκότητα $O(n^2)$:

Μπορούμε να χρησιμοποιήσουμε κάπως το πρόβλημα της Μέγιστης Κοινής Υπακολουθίας;

- Φτιάχνουμε την ακολουθία $a'_1, \ldots, a'_m$ η οποία αποτελείται από τα διαφορετικά στοιχεία της αρχικής ακολουθίας ταξινομημένα κατά αύξουσα σειρά.
- Η Μέγιστη Κοινή Υπακολουθία μεταξύ της ταξινομημένης και της αρχικής είναι και η ζητούμενη Μέγιστη Αύξουσα Υπακολουθία.

Μπορούμε να βρούμε κάτι πιο γρήγορο;

Μέγιστη Αύξουσα Υπακολουθία (DPV ch6.2)

Λύση σε χρόνο $O(n \log n)$:

Μέγιστη Αύξουσα Υπακολουθία (DPV ch6.2)

Λύση σε χρόνο $O(n \log n)$:

Ορίζουμε:

- $m(\ell) =$ δείκτης i τέτοιος ώστε το a_i να είναι το μικρότερο στοιχείο στο οποίο τελειώνει κάποια αύξουσα υπακολουθία μήκους ℓ

Μέγιστη Αύξουσα Υπακολουθία (DPV ch6.2)

Λύση σε χρόνο $O(n \log n)$:

Ορίζουμε:

- $m(\ell) =$ δείκτης i τέτοιος ώστε το a_i να είναι το μικρότερο στοιχείο στο οποίο τελειώνει κάποια αύξουσα υπακολουθία μήκους ℓ
- $p(i) =$ ο δείκτης του προτελευταίου στοιχείου της μέγιστης υπακολουθίας που τελειώνει στο a_i.

Μέγιστη Αύξουσα Υπακολουθία (DPV ch6.2)

Λύση σε χρόνο $O(n \log n)$:

Ορίζουμε:

- $m(\ell) =$ δείκτης i τέτοιος ώστε το a_i να είναι το μικρότερο στοιχείο στο οποίο τελειώνει κάποια αύξουσα υπακολουθία μήκους ℓ
- $p(i) =$ ο δείκτης του προτελευταίου στοιχείου της μέγιστης υπακολουθίας που τελειώνει στο a_i.

π.χ.
Αύξουσες Υπακολουθίες μήκους $\ell = 1$:
5,2,6,3,6,9 $\rightarrow a_{m(1)} = a_2 = 2$

Αύξουσες Υπακολουθίες μήκους $\ell = 2$:
5-6,5-6,5-9,2-6,2-3,2-6,2-9,6-9,3-9,6-9 $\rightarrow a_{m(2)} = a_4 = 3$

Μέγιστη Αύξουσα Υπακολουθία (DPV ch6.2)

Λύση σε χρόνο $O(n \log n)$:

- $m(\ell) =$ δείκτης i τέτοιος ώστε το a_i να είναι το μικρότερο στοιχείο στο οποίο τελειώνει κάποια αύξουσα υπακολουθία μήκους ℓ
- $p(i) =$ ο δείκτης του προτελευταίου στοιχείου της μέγιστης υπακολουθίας που τελειώνει στο a_i.

Παρατήρηση: Το $a_{m(1)}, a_{m(2)}, \ldots, a_{m(L_{\max})}$ είναι αύξουσα υπακολουθία. (Γιατί; Παρατηρήστε: Το $a_{m(1)}$ είναι το μικρότερο στοιχείο του πίνακα. Το $a_{m(2)}$, ως τελευταίο στοιχείο αύξουσας υπακολουθίας μήκους 2, είναι γνησίως μεγαλύτερο από τουλάχιστον ένα στοιχείο, έστω a_k. Άρα $a_{m(1)} \leq a_k < a_{m(2)}$.)

Μέγιστη Αύξουσα Υπακολουθία (DPV ch6.2)

Λύση σε χρόνο $O(n \log n)$:

Για κάθε θέση του πίνακα $i = 1, \ldots, n$:

- Ψάχνω το μεγαλύτερο μήκος υπακολουθίας $\ell \in [0, L_{\max}]$ τέτοιο ώστε το $a_{m(\ell)} < a_i$ (δυαδική αναζήτηση στα $a_{m(1)}, a_{m(2)}, \ldots$).

Μέγιστη Αύξουσα Υπακολουθία (DPV ch6.2)

Λύση σε χρόνο $O(n \log n)$:

Για κάθε θέση του πίνακα $i = 1, \ldots, n$:

- Ψάχνω το μεγαλύτερο μήκος υπακολουθίας $\ell \in [0, L_{\max}]$ τέτοιο ώστε το $a_{m(\ell)} < a_i$ (δυαδική αναζήτηση στα $a_{m(1)}, a_{m(2)}, \ldots$).
- Το μήκος της μακρύτερης υπακολουθίας που τελειώνει στο a_i θα είναι $\ell + 1$ και $m(\ell)$ το προτελευταίο στοιχείο της. Άρα:
  - $m(\ell + 1) = i$ (αν υπήρχε στοιχείο $< a_i$ στο οποίο τελειώνει κάποια υπακολουθία μήκους $\ell + 1$, τότε η αναζήτηση δε θα είχε σταματήσει στο ℓ)
  - $p(i) = m(\ell)$.

Μέγιστη Αύξουσα Υπακολουθία (DPV ch6.2)

Λύση σε χρόνο $O(n \log n)$:

Για κάθε θέση του πίνακα $i = 1, \ldots, n$:

- Ψάχνω το μεγαλύτερο μήκος υπακολουθίας $\ell \in [0, L_{\max}]$ τέτοιο ώστε το $a_{m(\ell)} < a_i$ (δυαδική αναζήτηση στα $a_{m(1)}, a_{m(2)}, \ldots$).
- Το μήκος της μακρύτερης υπακολουθίας που τελειώνει στο a_i θα είναι $\ell + 1$ και $m(\ell)$ το προτελευταίο στοιχείο της. Άρα:
  - $m(\ell + 1) = i$ (αν υπήρχε στοιχείο $< a_i$ στο οποίο τελειώνει κάποια υπακολουθία μήκους $\ell + 1$, τότε η αναζήτηση δε θα είχε σταματήσει στο ℓ)
  - $p(i) = m(\ell)$.
- Αν $\ell + 1 > L_{\max}$ τότε αυτό είναι το νέο $L_{\max}$.

Outline

Edit Distance

Είσοδος: Δύο ακολουθείες χαρακτήρων

Έξοδος: Ο ελάχιστος αριθμός ενεργειών που απαιτούνται προκειμένου να γίνουν οι ακολουθίες ίδιες, όπου διαθέσιμες ενέργειες είναι:

- Προσθήκη Χαρακτήρα
- Διαγραφή Χαρακτήρα
- Αντικατάσταση Χαρακτήρα

Edit Distance

Είσοδος: Δύο ακολουθείες χαρακτήρων

Έξοδος: Ο ελάχιστος αριθμός ενεργειών που απαιτούνται προκειμένου να γίνουν οι ακολουθίες ίδιες, όπου διαθέσιμες ενέργειες είναι:

- Προσθήκη Χαρακτήρα
- Διαγραφή Χαρακτήρα
- Αντικατάσταση Χαρακτήρα

S U N N Y

S N O W Y

Edit Distance

Είσοδος: Δύο ακολουθείες χαρακτήρων

Έξοδος: Ο ελάχιστος αριθμός ενεργειών που απαιτούνται προκειμένου να γίνουν οι ακολουθίες ίδιες, όπου διαθέσιμες ενέργειες είναι:

- Προσθήκη Χαρακτήρα
- Διαγραφή Χαρακτήρα
- Αντικατάσταση Χαρακτήρα

S U N N - Y

S - N O W Y

Edit Distance

Λύση

Έστω $E[i, j]$ το edit distance των ακολουθιών μέχρι και τους χαρακτήρες i, j των ακολουθιών. Τότε

$$E[i, 0] = i$$
$$E[0, j] = j$$
$$E[i, j] = \min \begin{cases} E[i-1, j] + 1 \\ E[i, j-1] + 1 \\ E[i-1, j-1] + diff(i, j) \end{cases}$$

Όπου $diff(i, j)$ είναι 0 αν οι χαρακτήρες i, j των ακολουθιών ταυτίζονται και 1 διαφορετικά.