Αν. Καθηγητής Π. Λουρίδας

Τμήμα Διοικητικής Επιστήμης και Τεχνολογίας

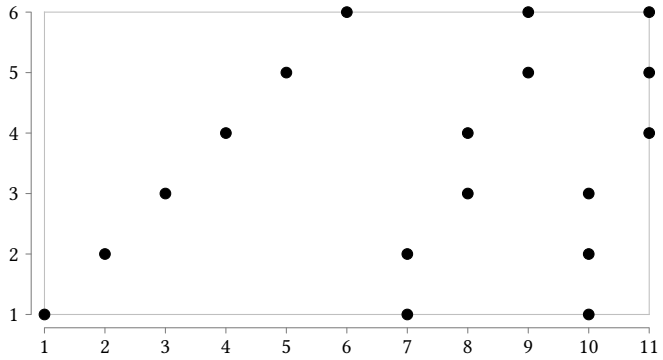
Οικονομικό Πανεπιστήμιο Αθηνών

Καλύπτοντας Σημεία

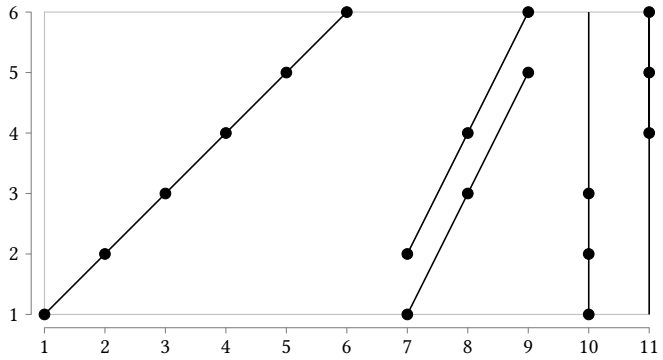
Φανταστείτε ότι έχετε ένα σύνολο σημείων στο επίπεδο και θέλετε να τα καλύψετε με ένα σύνολο ευθειών γραμμών. Μπορείτε να βρείτε τον ελάχιστο αριθμό ευθείων γραμμών που να περνάνε πάνω από τα σημεία ώστε να τα καλύψετε όλα;

Αυτό είναι το επονομαζόμενο πρόβλημα της στόχευσης σημείων (hitting objects problem), επειδή αντιστοιχεί σε ένα σύνολο αντικειμένων, που αναπαριστούμε με σημεία, τα οποία θέλουμε να τα πετύχουμε με όσο δυνατόν λιγότερες βολές γίνεται. Βεβαίως το πρόβλημα είναι πολύ γενικότερο και δεν περιορίζεται στη βαλιστική.

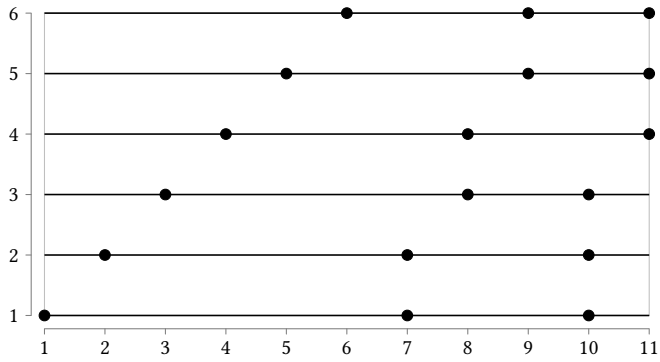
Για παράδειγμα, δείτε την παρακάτω εικόνα, η οποία δείχνει 18 σημεία.



Μπορούμε να καλύψουμε τα σημεία αυτά χρησιμοποιώντας πέντε γραμμές, όπως φαίνεται παρακάτω:



Εναλλακτικά, αν έχουμε ως επιπλέον απαίτηση οι γραμμές να είναι παράλληλες στους άξονες, τότε η βέλτιστη λύση χρησιμοποιεί έξι γραμμές:



Το ερώτημα είναι, πώς μπορούμε να βρούμε τις λύσεις αυτές;

Μέθοδος Επίλυσης

Για να λύσουμε το πρόβλημα, κατ' αρχήν θα πρέπει να βρούμε όλες τις γραμμές που περνούν μέσα από τα σημεία τα οποία δίνονται. Στη συνέχεια, θα πρέπει να επιλέξουμε τις λιγότερες δυνατές γραμμές που τα καλύπτουν όλα.

Η επιλογή των λιγότερων δυνατών γραμμών είναι ένα παράδειγμα του γενικότερου *προβλήματος κάλυψης συνόλου* (set covering problem), το οποίο ορίζεται ως εξής: αν έχουμε ένα σύνολο στοιχείων το οποίο καλούμε σύμπαν (universe) $U = \{1, 2, ... n\}$ και ένα σύνολο m συνόλων S των οποίων η ένωση είναι ίση με το σύμπαν, τότε να βρεθεί το ελάχιστο υποσύνολο του S του οποίου η ένωση να είναι ίση με το σύμπαν.

Για παράδειγμα, έστω ότι το σύμπαν μας είναι το $U = \{1, 2, 3, 4, 5\}$ και το σύνολο των συνόλων είναι το $S = \{\{1\}, \{1, 2\}, \{1, 2, 3\}, \{2, 4\}, \{3, 4\}, \{4, 5\}\}$. Τότε βλέπουμε ότι η ένωση των συνόλων που ανήκουν στο S μας δίνει το U, αλλά για να καλύψουμε το U αρκεί να χρησιμοποιήσουμε το υποσύνολο $\{\{1, 2, 3\}, \{4, 5\}\}$.

Δυστυχώς το πρόβλημα κάλυψης συνόλου δεν μπορεί να λυθεί σε πολυωνυμικό χρόνο. Πράγματι, σκεφτείτε την εξής προσέγγιση:

- Για κάθε δυνατό υποσύνολο του S:
  - Έλεγξε αν καλύπτει το U.
  - Αν ναι και είναι μικρότερο από την καλύτερη υποψήφια λύση που έχουμε βρει μέχρι τώρα, σημειώνουμε ότι αυτό αποτελεί πλέον την καλύτερη υποψήφια λύση.

Όταν η διαδικασία τελειώσει, η τελική υποψήφια λύση που θα έχουμε βρει θα είναι και η βέλτιστη. Το κακό βρίσκεται στη φράση «κάθε δυνατό υποσύνολο του S». Αν ένα σύνολο αποτελείται από m στοιχεία, ο αριθμός των δυνατών υποσυνόλων του είναι 2^m· επομένως, εκτός και αν το m είναι μικρό, δεν υπάρχει περίπτωση να βρούμε λύση σε εύθετο χρόνο.

Βεβαίως, αν το σκεφτείτε, δεν χρειάζεται να δοκιμάσει κάποιος κάθε δυνατό υποσύνολο του S. Αφού θέλουμε να βρούμε τον ελάχιστο αριθμό γραμμών, μπορούμε

βελτιώσουμε την προσέγγιση:

- Εξέτασε τα υποσύνολα του S σε αύξουσα σειρά μεγέθους (αριθμός ευθειών που περιέχουν):
  - Έλεγξε αν το τρέχον υποσύνολο καλύπτει το U.
  - Αν ναι, είναι η λύση που αναζητάμε και δεν χρειάζεται να συνεχίσουμε.

Αυτό θα μας βρει τη βέλτιστη λύση, αφού αν συνεχίσουμε μπορεί να βρούμε μια άλλη λύση που θα προκύπτει από υποσύνολο με τον ίδιο αριθμό ευθειών, που δεν χρειάζεται, ή με μεγαλύτερο αριθμό ευθειών, που δεν το θέλουμε.

Το θέμα είναι ότι ούτε αυτή η βελτιωμένη προσέγγιση τρέχει πάντα σε κάποιον αποδεκτό χρόνο. Μπορεί τα σημεία να είναι τοποθετημένα έτσι ώστε μόνο ανά δύο να είναι συγγραμμικά. Τότε, αν έχουμε n σημεία, απαιτούνται $n/2$ ευθείες. Εμείς θα ξεκινήσουμε να ελέγχουμε με μικρότερο αριθμό ευθειών, για να φτάσουμε τελικά στον μέγιστο αριθμό των $n/2$ ευθειών, δηλαδή:

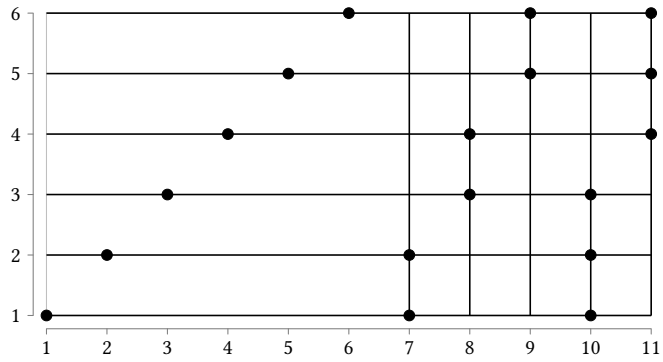
- Υποσύνολα με μηδέν ευθείες (το κενό υποσύνολο).
- Υποσύνολα με μία ευθεία.
- Υποσύνολα με δύο ευθείες.
- Κ.ο.κ. μέχρι να φτάσουμε στα υποσύνολα με $n/2$ ευθείες.

Επομένως, τελικά, θα εξετάσουμε πάλι έναν εκθετικό αριθμό υποσυνόλων, της τάξης του $2^{n/2}$. Μπορεί βεβαίως να παρατηρήσετε ότι τα υποσύνολα με μηδέν και μία ευθεία είναι περιττό να τα ελέγξουμε, αλλά πρακτικά ουδόλως μας επιβαρύνουν μπροστά στο πλήθος των περιπτώσεων που θα πρέπει να ελέγξουμε. Αν θέλετε να δείτε ένα παράδειγμα σημείων που μόνο ανά δύο είναι συγγραμικά, μπορείτε να ανατρέξετε στη σελίδα https://en.wikipedia.org/wiki/No-three-in-line_problem.

Εναλλακτικά, μπορούμε να χρησιμοποιήσουμε μια άλλη προσέγγιση, η οποία θα ολοκληρωθεί γρήγορα, αλλά δεν μπορεί να μας εγγυηθεί ότι θα βρει τη βέλτιστη λύση:

- Όσο δεν έχουν καλυφθεί όλα τα στοιχεία του U:
  - Βρες το σύνολο του S το οποίο καλύπτει τον μέγιστο αριθμό στοιχείων που δεν έχουν καλυφθεί και πρόσθεσέ το στη λύση.

Η προσέγγιση αυτή είναι ένας *άπληστος αλγόριθμος* (greedy algorithm), επειδή κάθε φορά προχωράει με γνώμονα το μέγιστο άμεσο κέρδος (τον μέγιστο αριθμό στοιχείων που μπορούν να καλυφθούν). Δεν είναι όμως βέβαιο ότι όταν τελειώσουμε θα έχουμε βρει πράγματι την καλύτερη δυνατή λύση. Αν επιστρέψουμε στο παράδειγμα με τα σημεία και τις γραμμές παράλληλες στους άξονες, ένας άπληστος αλγόριθμος θα μπορούσε να καταλήξει στην παρακάτω λύση, αντί για τη βέλτιστη:



Γιατί; Διότι εργαζόμενος άπληστα, ο αλγόριθμος μπορεί να επιλέξει αρχικά την κάθετη ευθεία που τέμνει τον οριζόντιο άξονα στο 11, μετά την κάθετη που τέμνει τον οριζόντιο άξονα στο 10, θα συνεχίσει έτσι προσθέτοντας κάθετες ευθείες μέχρι και το σημείο 7 του οριζόντιου άξονα, και στη συνέχεια θα πρέπει να καλύψει τα σημεία με τετμημένες από 1 έως και 6, το οποίο θα κάνει προσθέτοντας τις οριζόντιες ευθείες του σχήματος.

Απαιτήσεις Προγράμματος

Κάθε φοιτητής θα εργαστεί σε αποθετήριο στο GitHub. Για να αξιολογηθεί μια εργασία θα πρέπει να πληροί τις παρακάτω προϋποθέσεις:

- Για την υποβολή της εργασίας θα χρησιμοποιηθεί το ιδιωτικό αποθετήριο του φοιτητή που δημιουργήθηκε για τις ανάγκες του μαθήματος και του έχει αποδοθεί. Το αποθετήριο αυτό έχει όνομα του τύπου `username-algo-assignments`, όπου `username` είναι το όνομα του φοιτητή στο GitHub. Για παράδειγμα, το σχετικό αποθετήριο του διδάσκοντα θα ονομαζόταν `louridas-algo-assignments` και θα ήταν προσβάσιμο στο https://github.com/dmst-algorithms-course/louridas-algo-assignments. *Τυχόν άλλα αποθετήρια απλώς θα αγνοηθούν.*
- Μέσα στο αποθετήριο αυτό θα πρέπει να δημιουργηθεί ένας κατάλογος `assignment-2021-2`.
- Μέσα στον παραπάνω κατάλογο το πρόγραμμα θα πρέπει να αποθηκευτεί με το όνομα `points_cover.py`.
- Δεν επιτρέπεται η χρήση έτοιμων βιβλιοθηκών γράφων ή τυχόν έτοιμων υλοποιήσεων των αλγορίθμων, ή τμημάτων αυτών, εκτός αν αναφέρεται ρητά ότι επιτρέπεται.
- Επιτρέπεται η χρήση δομών δεδομένων της Python όπως στοίβες, λεξικά, σύνολα, κ.λπ.
- Επιτρέπεται η χρήση της βιβλιοθήκης `itertools`.
- Επιτρέπεται η χρήση της βιβλιοθήκης `argparse` ή της βιβλιοθήκης `sys` (συ-

γκεκριμένα, της λίστας `sys.argv`) προκειμένου να διαβάσει το πρόγραμμα τις παραμέτρους εισόδου.

- Το πρόγραμμα θα πρέπει να είναι γραμμένο σε Python 3.

Το πρόγραμμα θα καλείται ως εξής (όπου `python` η κατάλληλη εντολή στο εκάστοτε σύστημα):

```
python points_cover.py [-f] [-g] points_file
```

Η σημασία των παραμέτρων είναι η εξής:

- Η παράμετρος `-f` είναι προαιρετική. Αν δίνεται, το πρόγραμμα θα βρει τη βέλτιστη λύση, εξετάζοντας όσα υποσύνολα χρειαστεί. Αν οι γραμμές δεν είναι λίγες, το πρόγραμμα δεν θα προλάβει να τελειώσει σε λογικό χρόνο, αλλά αυτό είναι πρόβλημα του χρήστη (ας πρόσεχε). Διαφορετικά, αν ο χρήστης δεν δώσει την παράμετρο `-f`, θα εκτελεστεί ο άπληστος αλγόριθμος.
- Η παράμετρος `-g` είναι προαιρετική. Αν δίνεται, το πρόγραμμα θα αναζητήσει μόνο γραμμές που είναι παράλληλες με τους άξονες. Διαφορετικά, μπορεί να χρησιμοποιήσει οποιεσδήποτε γραμμές περνούν από τα σημεία.
- `points_file`: το αρχείο που περιέχει τα σημεία που πρέπει να καλυφθούν. Το αρχείο θα αποτελείται από γραμμές της μορφής:

  ```
  x y
  ```

  όπου `x` είναι η x και `y` είναι η y συντεταγμένη κάθε σημείου.

Στην περίπτωση που δοθεί η παράμετρος `-g` και δεν αρκούν οι γραμμές που συνδέουν τα σημεία που δίνονται για να γίνει πλήρης κάλυψη, μπορείτε να χρησιμοποιήσετε και οριζόντιες γραμμές που περνούν από τα σημεία τα οποία δεν είναι συγγραμμικά με άλλο οριζόντιο ή κάθετο σημείο. Αν λοιπόν υπάρχει σημείο (x, y) και δεν υπάρχει σημείο (x', y) ή (x, y'), μπορείτε να χρησιμοποιήσετε τη γραμμή $((x, y), (x + 1, y))$.

Οι ευθείες που θα απαρτίζουν τις λύσεις που βρίσκετε θα πρέπει να εμφανίζονται με φθίνουσα σειρά αριθμών σημείων που καλύπτουν, δηλαδή οι ευθείες που καλύπτουν περισσότερα σημεία θα προηγούνται αυτών που καλύπτουν λιγότερα σημεία. Σε κάθε μία ευθεία, τα σημεία θα εμφανίζονται σε αύξουσα σειρά σύμφωνα με τις συντεταγμένες τους. Οι ευθείες που καλύπτουν τον ίδιο αριθμό σημείων θα εμφανίζονται σε αύξουσα σειρά σύμφωνα με τις συντεταγμένες των σημείων τους.

Προκειμένου να μπορέσετε να αναπαράξετε τα αποτελέσματα που δίνονται στα παραδείγματα που ακολουθούν μπορεί να χρειαστεί να ταξινομήσετε τα σημεία που θα διαβάσετε από τα αρχεία των παραδειγμάτων.

Παραδείγματα

Παράδειγμα 1

Αν ο χρήστης του προγράμματος δώσει:

```
python points_cover.py example_1.txt
```

τότε το πρόγραμμα θα διαβάσει το αρχείο σημείων `example_1.txt` και θα πρέπει να εμφανίσει στην οθόνη *ακριβώς* τα παρακάτω (και τίποτε παραπάνω):

```
(1, 1) (2, 2) (3, 3) (4, 4) (5, 5) (6, 6)
(7, 1) (8, 3) (9, 5)
(7, 2) (8, 4) (9, 6)
(10, 1) (10, 2) (10, 3)
(11, 4) (11, 5) (11, 6)
```

Αυτή είναι η πρώτη λύση που είδαμε για τα 18 σημεία.

Παράδειγμα 2

Αν ο χρήστης του προγράμματος δώσει:

```
python points_cover.py -f -g example_1.txt
```

τότε το πρόγραμμα θα διαβάσει το αρχείο σημείων `example_1.txt` και θα πρέπει να εμφανίσει στην οθόνη *ακριβώς* τα παρακάτω (και τίποτε παραπάνω):

```
(1, 1) (7, 1) (10, 1)
(2, 2) (7, 2) (10, 2)
(3, 3) (8, 3) (10, 3)
(4, 4) (8, 4) (11, 4)
(5, 5) (9, 5) (11, 5)
(6, 6) (9, 6) (11, 6)
```

Αυτή είναι η δεύτερη λύση που είδαμε για τα 18 σημεία.

Παράδειγμα 3

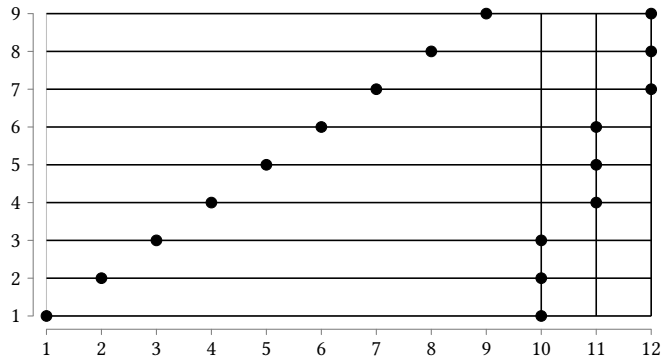
Αν ο χρήστης του προγράμματος δώσει:

```
python points_cover.py -g example_2.txt
```

τότε το πρόγραμμα θα διαβάσει το αρχείο σημείων `example_2.txt` και θα πρέπει να εμφανίσει στην οθόνη *ακριβώς* τα παρακάτω (και τίποτε παραπάνω):

```
(10, 1) (10, 2) (10, 3)
(11, 4) (11, 5) (11, 6)
(12, 7) (12, 8) (12, 9)
(1, 1) (10, 1)
(2, 2) (10, 2)
(3, 3) (10, 3)
(4, 4) (11, 4)
(5, 5) (11, 5)
```

```
(6, 6) (11, 6)
(7, 7) (12, 7)
(8, 8) (12, 8)
(9, 9) (12, 9)
```



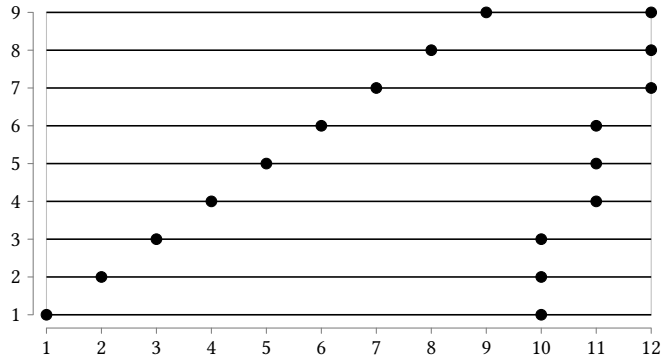
Παράδειγμα 4

Αν ο χρήστης του προγράμματος δώσει:

```
python points_cover.py -f -g example_2.txt
```

τότε το πρόγραμμα θα διαβάσει το αρχείο σημείων `example_2.txt` και θα πρέπει να εμφανίσει στην οθόνη *ακριβώς* τα παρακάτω (και τίποτε παραπάνω):

```
(1, 1) (10, 1)
(2, 2) (10, 2)
(3, 3) (10, 3)
(4, 4) (11, 4)
(5, 5) (11, 5)
(6, 6) (11, 6)
(7, 7) (12, 7)
(8, 8) (12, 8)
(9, 9) (12, 9)
```



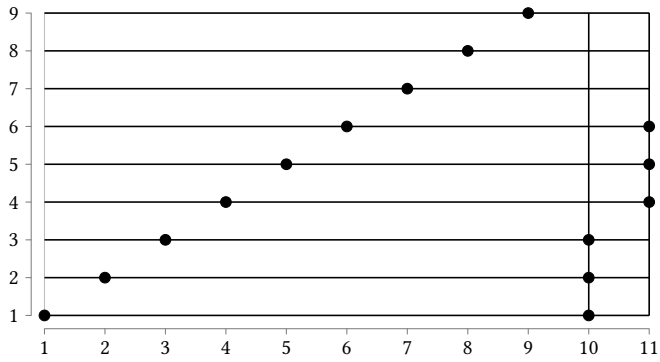
Παράδειγμα 5

Αν ο χρήστης του προγράμματος δώσει:

```
python points_cover.py -g example_3.txt
```

τότε το πρόγραμμα θα διαβάσει το αρχείο σημείων `example_3.txt` και θα πρέπει να εμφανίσει στην οθόνη *ακριβώς* τα παρακάτω (και τίποτε παραπάνω):

```
(10, 1) (10, 2) (10, 3)
(11, 4) (11, 5) (11, 6)
(1, 1) (10, 1)
(2, 2) (10, 2)
(3, 3) (10, 3)
(4, 4) (11, 4)
(5, 5) (11, 5)
(6, 6) (11, 6)
(7, 7) (8, 7)
(8, 8) (9, 8)
(9, 9) (10, 9)
```



Παράδειγμα 6

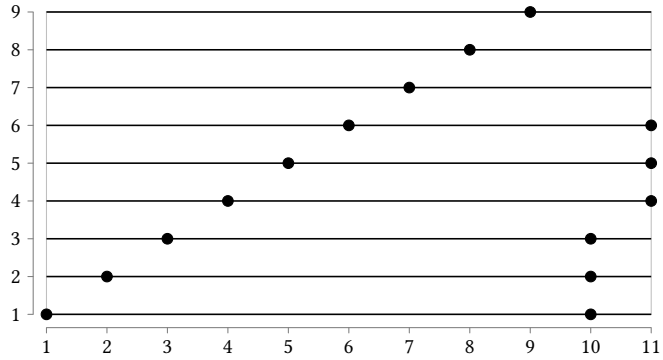
Αν ο χρήστης του προγράμματος δώσει:

```
python points_cover.py -f -g example_3.txt
```

τότε το πρόγραμμα θα διαβάσει το αρχείο σημείων `example_3.txt` και θα πρέπει να εμφανίσει στην οθόνη *ακριβώς* τα παρακάτω (και τίποτε παραπάνω):

```
(1, 1) (10, 1)
(2, 2) (10, 2)
(3, 3) (10, 3)
(4, 4) (11, 4)
(5, 5) (11, 5)
(6, 6) (11, 6)
(7, 7) (8, 7)
```

```
(8, 8) (9, 8)
(9, 9) (10, 9)
```



Καλή Επιτυχία.