Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

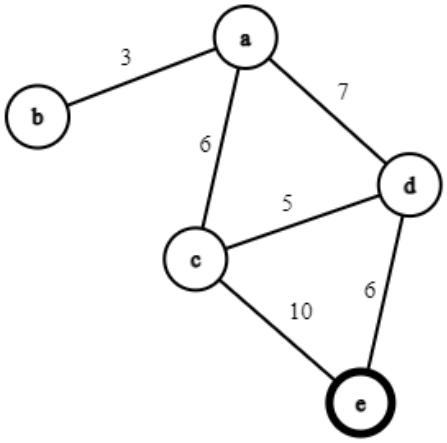
Αλγόριθμοι και Πολυπλοκότητα

3η Σειρά Γραπτών Ασκήσεων – Χειμερινό Εξάμηνο 2021

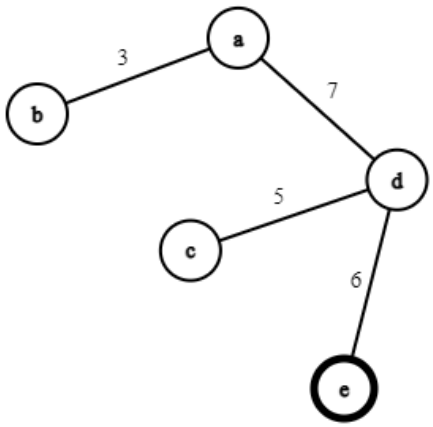
Κονταλέξη Μαρίνα – el18022

Άσκηση 1:

(α) Εύκολα βρίσκουμε αντιπαράδειγμα για τον ισχυρισμό.
Στον παρακάτω γράφο G ζητάμε το T*(a, 2).

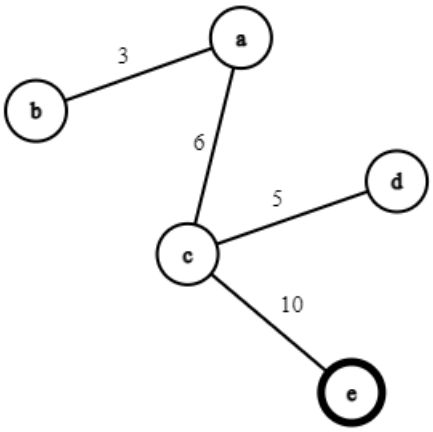


Το ελάχιστο συνδετικό δέντρο T*(a,2) που περιέχει τουλάχιστον 2 από τις ακμές του a είναι το παρακάτω:



Παρατηρούμε πως το T*(a,2) δεν περιέχει τις 2 ελαφρύτερες ακμές του κόμβου a.

Υπολογίζοντας το ελάχιστο συνδετικό δέντρο που περιέχει τις 2 ελαφρύτερες ακμές του κόμβου a, θα καταλήξουμε στο παρακάτω που είναι βαρύτερο από το T*(a, 2).



Άρα, δείξαμε πως το T*(s, k) δεν περιέχει πάντα τις k ελαφρύτερες ακμές του α.

(β) Η ιδέα του αλγορίθμου για να υπολογίσουμε το T*(s, k) είναι να τρέχουμε διαδοχικές φορές τον αλγόριθμο του Kruskal για την κατασκευή ΕΣΔ, κάνοντας κάθε φορά αλλαγές στην ταξινόμηση των ακμών. Οι αλλαγές αποσκοπούν στην:

- «προτεραιοποίηση» των ακμών του κόμβου s όταν το προηγούμενο ΕΣΔ περιέχει λιγότερες από τις k επιθυμητές
- «αποπροτεραιοποίηση» των ακμών του κόμβου s όταν το προηγούμενο ΕΣΔ περιέχει περισσότερες από τις k επιθυμητές

Η αυξομείωση της προτεραιότητας αφορά το βάρος των ακμών, εφόσον κριτήριο του Kruskal για να εξετάσει μία ακμή νωρίτερα από κάποια άλλη (και εάν είναι αποδεκτή να την προσθέσει στο ΕΣΔ) είναι να είναι ελαφρύτερη. Επομένως, επιλέγουμε να προσθέτουμε ή να αφαιρούμε κατάλληλη μεταβλητή p στα βάρη των ακμών μέχρι να πετύχουμε τον στόχο k.

Με προφανή τρόπο καταλαβαίνουμε πως (δεδομένου ότι το T*(s,k) υπάρχει) μπορούμε προσθέτοντας $p = E_{max}$ να «μεταφέρουμε» όλες τις ακμές στο τέλος της ταξινομημένης λίστας που εξετάζει ο αλγόριθμος μας, και άρα να πετύχουμε την απαίτηση k = 1, ενώ αφαιρώντας $p = E_{max}$ τότε οι k = deg(s) πρώτες ακμές που θα προστεθούν στο ΕΣΔ είναι οι ακμές που προσπίπτουν στον κόμβο s (διότι είναι οι ελαφρύτερες και δεν δημιουργούν κύκλο), με αποτέλεσμα να ικανοποιείται η απαίτηση k = deg(s). Κάνοντας δυαδική αναζήτηση για την τιμή της μεταβλητής p στο διάστημα $[-E_{max}, E_{max}]$, μπορούμε να επιτύχουμε κάθε απαίτηση ακμών k, δοκιμάζοντας να επανυπολογίσουμε το ΕΣΔ για κάθε p. Εάν ο βαθμός της s στο ΕΣΔ είναι $\deg_{MST}(s) > k$, τότε συνεχίζουμε τη δυαδική αναζήτηση στο μεγαλύτερο μισό του διαστήματος, εάν ισχύει $\deg_{MST}(s) < k$ τότε συνεχίζουμε τη δυαδική αναζήτηση στο μεγαλύτερο μισό του διαστήματος ενώ εάν $\deg_{MST}(s) = k$ τότε βρήκαμε το ζητούμενο MST.

Η υπολογιστική πολυπλοκότητα του αλγορίθμου μας είναι O(|E|log|V|) για κάθε φορά που τρέχουμε τον αλγόριθμο του Kruskal. Στη χειρότερη περίπτωση θα γίνουν $\log(2E_{\max})$ επαναλήψεις. Επομένως, η συνολική πολυπλοκότητα που προκύπτει είναι $O(|E| \log|V| \log(E_{max}))$

Άσκηση 2:

1. Δημιουργούμε ένα κατευθυνόμενο γράφημα G' για το οποίο θα ισχύει:

$$\begin{cases} \text{ακμές } V' = V \\ \text{κορυφές } E' = E \\ \text{βάρη p': } E \to \mathbb{Z} \mid p'(u,v) = p(v) \end{cases}$$

Δηλαδή, μεταφέρουμε το βάρος κάθε κορυφής στην ακμή με την οποία φτάσαμε στην κορυφή αυτή. Προφανώς, δεν μπορούμε ούτε να μην επιβαρυνθούμε το βάρος της κορυφής, καθώς το βάρος της ακμής μέσω της οποία φτάσαμε εκεί θα έχει προστεθεί στο βάρος του μονοπατιού, ούτε να επιβαρυνθούμε παραπάνω από μία φορά, καθώς η εξερχόμενη από την κορυφή ακμή θα αφορά την «συνέπεια» που θα έχει ο παίκτης από την επόμενη κορυφή στην οποία θα βρεθεί.

Επίσης, προσθέτουμε μία κορυφή s_0 στο γράφημα και μία ακμή (s_0, s), με $p(s_0, s) = r$.

Το πρόβλημα είναι ισοδύναμο με το να αναζητήσουμε μονοπάτι από την κορυφή s_0 στην κορυφή t, προσέχοντας να μη φτάσουμε σε κανέναν κόμβο του μονοπατιού με κόστος μη θετικό, διότι αυτό συνεπάγεται ότι ο παίκτης χάνει τη ζωή του.

Το πρόβλημα προσδιορίζει πως δεν υπάρχουν κύκλοι θετικού μήκους, άρα μπορούμε να χρησιμοποιήσουμε την παρακάτω τροποποιημένη μορφή του Bellman Ford.

Αναζητούμε τα μεγαλύτερα μονοπάτια μεταξύ του s_0 και των υπόλοιπων κόμβων, άρα θα αρχικοποιήσουμε $D[u] \leftarrow -\infty$ και η συνθήκη ανανέωσης των αποστάσεων θα είναι $D[u] = \max_{(v,u)\in p'} \big(D[u], D[v] + p'(v,u)\big)$, $\boldsymbol{if\ D[v] + p'(v,u) > 0}$.

Η συνθήκη $D[v] + p'(v,u) \geq 0$ είναι εκείνη που εξασφαλίζει πως δεν θα χρησιμοποιήσουμε ποτέ σαν ενδιάμεσο κόμβο κάποιον κόμβο u για τον οποίο $D[u] < 0$, επομένως, δεν θα μηδενιστεί ποτέ η ζωή του παίκτη κατά τη διαδρομή που βρίσκει ο Bellman Ford. Αν κατά τον τερματισμό έχουμε $D[t] = -\infty$ τότε συμπεραίνουμε πως κανένα μονοπάτι δεν είναι r-ασφαλές.

Εξαιτίας του παραπάνω, όταν υπάρχει λύση του προβλήματος ο αλγόριθμος βρίσκει πάντα ορθό μονοπάτι (δηλαδή μονοπάτι με δύναμη ≥ 0) , ενώ όταν δεν υπάρχει λύση το αντιλαμβανόμαστε από την τιμή $D[t] = -\infty$.

Η υπολογιστική πολυπλοκότητα που απαιτείται είναι O(|V| + |E|) για την κατασκευή του G' και O(|V||E|) για τον BF.

2. Εκτελούμε τον ίδιο αλγόριθμο με το ερώτημα 1, εκτελώντας τον BF μία ακόμη φορά.

Αν ο αλγόριθμος έχει ήδη βρει μονοπάτι μέχρι τον κόμβο t, τότε λαμβάνουμε θετική απάντηση όμοια με το ερώτημα 1.

Αν ο αλγόριθμος δεν έχει βρει μονοπάτι τότε κατά τη |V|-οστή επανάληψη του BF μπορούμε να εντοπίσουμε εάν οι κόμβοι που είναι προσβάσιμοι από τον s_0 (ισοδύναμα από την s) συμμετέχουν σε κύκλο θετικού μήκους, εφόσον μόνο τότε θα παρατηρηθεί αλλαγή στην τιμή κάποιου $D[u]$.

- Αν δεν παρατηρηθεί αλλαγή, τότε κανένας κύκλος θετικού μήκους δεν είναι προσβάσιμος από τον κόμβο s, άρα η απάντηση στο πρόβλημα είναι πως δεν υπάρχει r-ασφαλές μονοπάτι.
- Αν παρατηρηθεί αλλαγή, τότε κάθε κόμβος u με αυξημένο D[u] είτε συμμετέχει είτε έπεται κάποιου κύκλου θετικού μήκους. Άρα, αρκεί να κάνουμε DFS από κάθε κόμβο που άλλαξε ώστε να εξετάσουμε εάν ο κόμβος t είναι προσβάσιμος από κάποιον κύκλο θετικού μήκους. Εάν κάποιο από τα DFS βρει μονοπάτι μέχρι τον κόμβο t, τότε το μονοπάτι s-t είναι r-ασφαλές για κάθε r. Αντίθετα, αν κανένα από τα DFS δεν βρει μονοπάτι μέχρι τον t, τότε ο κόμβος δεν είναι προσβάσιμος από κύκλο θετικού μήκους και το μονοπάτι παραμένει όχι r-ασφαλές.

Συνολικά, ο αλγόριθμος έχει υπολογιστική πολυπλοκότητα $O(|V||E|)$ εξαιτίας του ερωτήματος 1 και $O(|V| + |E|)$ για κάθε εκτέλεση του DFS. Μπορεί να χρειαστούν το πολύ |V| εκτελέσεις DFS, άρα έχουμε $\mathrm{O}\big(|\mathrm{V}||\mathrm{E}| + |\mathrm{V}| * (|\mathrm{V}| + |\mathrm{E}|)\big) = \mathrm{O}(|\mathrm{V}||\mathrm{E}| + |\mathrm{V}|^2)$

3. Θα μετατρέψουμε το πρόβλημα απόφασης του ερωτήματος 2 σε πρόβλημα βελτιστοποίησης, χρησιμοποιώντας δυαδική αναζήτηση. Θα δοκιμάσουμε να λύσουμε το πρόβλημα 2 για τιμές r στο διάστημα $(0, \mathrm{N}_{\max}]$, όπου $\mathrm{N}_{\max} = \sum_{u,v} p'(u,v)\,, p'(u,v) \leq 0$.
   Όταν ο αλγόριθμος δεν βρίσκει λύση για κάποιο r, συνεχίζουμε τη δυαδική αναζήτηση στο μεγαλύτερο μισό του διαστήματος και όταν ο αλγόριθμος βρίσκει λύση για κάποιο r, συνεχίζουμε τη δυαδική αναζήτηση στο μικρότερο μισό του διαστήματος, μέχρι να φτάσουμε στην οριακή τιμή του r.
   Συνολική πολυπλοκότητα του αλγορίθμου είναι $O\big(\log(N_{max})\,(|V||E| + |V|^2)\big)$.

Άσκηση 3:

1. Για την επίλυση του προβλήματος θα χρειαστούμε τον πίνακα $p(i) = \sum_1^i b(u_1, u_k)$ που περιέχει την πληροφορία της απόστασης ενός κόμβου i από τον κόμβο s σε λίτρα βενζίνης. Ο πίνακας αποτελεί τον πίνακα prefix sum των τιμών b.

   Η αναλλοίωτη του αλγορίθμου είναι πως φτάνουμε σε κάθε κόμβο u με βενζίνη b τέτοια ώστε:
   I. $b = 0$: αν ο κόμβος j από τον οποίο ανεφοδιαστήκαμε τελευταία φορά είχε τιμή $c(j)$: $c(j) > c(u)$ αλλιώς
   II. $b \geq 0$: αν ο κόμβος j από τον οποίο ανεφοδιαστήκαμε τελευταία φορά είχε τιμή $c(j)$: $c(j) \leq c(u)$

   Αν βρεθούμε σε έναν κόμβο u, τότε η βέλτιστη λύση είναι η εξής:

   - Αν *δεν* μπορούμε να φτάσουμε σε κόμβο j με $c(j) < c(u)$ χρησιμοποιώντας βενζίνη B, τότε βρίσκουμε την πόλη w με την φθηνότερη τιμή $c(w)$ σε διάστημα προσβάσιμο με ντεπόζιτο μεγέθους B. Γεμίζουμε πλήρως το ντεπόζιτο και συνεχίζουμε από την πόλη w. Ο λόγος που γεμίζουμε πλήρως το ντεπόζιτο είναι πως χρειαζόμαστε ανεφοδιασμό μέχρι να διασχίσουμε ολόκληρο το διάστημα $[p(u), p(u) + B]$, και ο φθηνότερος σταθμός ανεφοδιασμού είναι αυτός στον οποίο ήδη βρισκόμαστε. Άρα, καλύπτουμε τις ανάγκες για το διάστημα p(u) + B και συνεχίζουμε μέχρι τον αμέσως φθηνότερο. Η συνθήκη II της αναλλοίωτης εκπληρώνεται.
   - Αν μπορούμε να φτάσουμε σε κόμβο j με $c(j) < c(i)$ χρησιμοποιώντας βενζίνη B, τότε επιλέγουμε να προσθέσουμε στο ντεπόζιτό μας ακριβώς όση βενζίνη μας υπολείπεται για να φτάσουμε στον κόμβο j. Ο λόγος που το επιλέγουμε αυτό είναι για να ανεφοδιαστούμε όσο φθηνότερα γίνεται μέχρι και τον κόμβο j και από εκεί να συνεχίσουμε το ταξίδι μας χρησιμοποιώντας φθηνότερη βενζίνη από αυτή στην οποία έχουμε πρόσβαση αυτή τη στιγμή. Η συνθήκη I της αναλλοίωτης εκπληρώνεται.

   Ο λόγος που σε καμία από τις παραπάνω περιπτώσεις δεν ασχολούμαστε με την βενζίνη με την οποία φτάνουμε στον κόμβο u είναι πως λόγω της αναλλοίωτης, αν φτάσαμε στον κόμβο u με περισσευούμενη βενζίνη, τότε αυτή δεν μας φτάνει ώστε να φτάσουμε σε έναν κόμβο j με $j > u$ τέτοιο ώστε $c(j) < c(u)$ καθώς δεν υπάρχει τέτοιος κόμβος στο διάστημα που μπορούμε να διασχίσουμε με το παρόν ντεπόζιτο.

   Άρα, αρκεί να βρίσκουμε κάθε φορά τον επόμενο κόμβο j από τον οποίο θα ανεφοδιαστούμε. Στην πρώτη περίπτωση θα είναι ο πρώτος κόμβος j τέτοιος ώστε $c(j) < c(u)$ *και* $p(j) - p(u) \leq B$. Ονομάζουμε τον πρώτο όρο j για τον οποίο ισχύει $c(j) < c(u)$ *και* $p(j) - p(u) \geq 0$ *ελάσσονα όρο* του u και συμβολίζουμε $j = E_{min}(u)$. Στη δεύτερη περίπτωση θα είναι ο κόμβος j με την μικρότερη τιμή $c(j)$ για τον οποίο ισχύει $p(j) \leq B$ τον οποίο συμβολίζουμε $j = M(u)$. Για τον υπολογισμό τους θα χρειαστούμε τη βοηθητική συνάρτηση next(i) καθώς και τις δομές MH που είναι ένας σωρός ελαχίστου και A που είναι ένας πίνακας ακεραίων. Τις ορίζουμε παρακάτω:

   - Κατασκευάζουμε τον πίνακα A χρησιμοποιώντας τον αλγόριθμο της άσκησης 3 της πρώτης σειράς ασκήσεων, ο οποίος υπολογίζει τον πρώτο μεγαλύτερο όρο αριστερότερα από έναν όρο i. Αντιστρέφοντας τον πίνακα c και χρησιμοποιώντας τις τιμές $-c_i$ υπολογίζουμε το ζητούμενο σε O(|V|) και προκύπτει ο πίνακας E_{min}.
   - Θα χρησιμοποιήσουμε τη δομή MH για να υπολογίσουμε αποδοτικά τον πίνακα M. Θα χρησιμοποιήσουμε window sliding για να υπολογίσουμε το ζητούμενο για όλα τα διαστήματα μήκους B. Κατασκευάζουμε τον σωρό MH ως εξής:

1) Διατρέχουμε τον πίνακα c. Ξεκινάμε με start = 1 και $current = 1$
2) Αν $p(u_{current}) - p(u_{start}) \leq B$ τότε προσθέτουμε το $(c_{current}, current)$ στον σωρό ελαχίστου MH (τον οποίο ταξινομούμε με βάση το πρώτο στοιχείο του tuple). Συνεχίζουμε στο βήμα 7.
3) Βρίσκουμε το $\min(MH) = (c_m, position_m)$.
4) Αν $position_m > start$ τότε θέτουμε $M[start] = c_m$ και αφαιρούμε το min(MH) από τον σωρό. Συνεχίζουμε στο βήμα 6.
5) Αν $position_m \leq start$ τότε αφαιρούμε το min(MH) από τον σωρό (καθώς έχει μείνει στον σωρό κάποιο στοιχείο εκτός του παραθύρου B που εξετάζουμε) και επιστρέφουμε στο βήμα 3.
6) Θέτουμε start = start + 1. Αν start == n τότε τερματίζουμε.
7) Αν $current < n - 1$ θέτουμε current = current + 1 και επιστρέφουμε στο βήμα 2.
8) Αλλιώς συνεχίζουμε από το βήμα 3.

Η υπολογιστική πολυπλοκότητα του παραπάνω είναι $O(|V|\log |V|)$ καθώς κάνουμε |V| προσθήκες και το πολύ |V| αφαιρέσεις ελαχίστου από τον σωρό καθεμία από τις οποίες κοστίζει $O(\log|V|)$. Επίσης ζητάμε |V| φορές το ελάχιστο του σωρού, το οποίο προσθέτει πολυπλοκότητα $O(|V|)$. Συνολικά έχουμε $O(|V|\log|V|)$.

- Τέλος ορίζουμε τη συνάρτηση $next(i, b) = (next_{node}, b_{new})$.

$$next(i,b) = \begin{cases} (E_{min}[i], b'), & \alpha\nu\ p(E_{min}[i]) - p(i) \leq B \\ (M[i], B - b), & \alpha\lambda\lambda\iota\acute{\omega}\varsigma \end{cases}$$

Όπου $b' = p(E_{min}[i]) - p(i) - b$

Τα βήματα του αλγορίθμου φαίνονται παρακάτω:
1) Βρισκόμαστε στον κόμβο $u_i = u_2$ με αρχικό κόστος $c = 0$ και αρχική βενζίνη $b = 0$
2) Θέτουμε $(k, b') = next(i, b)$ και αυξάνουμε $c \mathrel{+}= c(u_i) * b'$, όπου το γινόμενο προκύπτει από την βενζίνη που αγοράσαμε από τον κόμβο u_i πολλαπλασιασμένη με την τιμή στην οποία την αγοράσαμε.
3) Εάν $k == n$ τερματίζουμε, αλλιώς επαναλαμβάνουμε τα βήματα 2, 3 με $i = k$.

Συνολικά, έχουμε υπολογιστική πολυπλοκότητα $O(|V|)$ για τους πίνακες p, E_{min}, next και $O(|V|\log|V|)$ για τον υπολογισμό του M, δηλαδή $O(|V|\log|V|)$.

2. Θα χρησιμοποιήσουμε δυναμικό προγραμματισμό, κρατώντας το σκεπτικό του ερωτήματος 1 σχετικά με το πότε γεμίζουμε το ντεπόζιτο (όταν πρόκειται να μεταβούμε σε κόμβο με μεγαλύτερη τιμή από αυτόν που βρισκόμαστε) και πότε μεταβαίνουμε σε έναν κόμβο χρησιμοποιώντας ακριβώς τόση βενζίνη όση χρειαζόμαστε για να διανύσουμε τη δεδομένη απόσταση (όταν πρόκειται να μεταβούμε σε κόμβο με μικρότερη τιμή από αυτόν που βρισκόμαστε).
Εφόσον θα χρησιμοποιήσουμε dp δεν μπορούμε να χρησιμοποιήσουμε το greedy κριτήριο της συνάρτησης next και έτσι θα πρέπει να ελέγχουμε τη μετάβαση σε κάθε πιθανό ενδιάμεσο κόμβο.
Η αναδρομική σχέση φαίνεται παρακάτω όπου ορίζουμε $d(u, b)$ το ελάχιστο κόστος να μεταβούμε στην κορυφή u ξεκινώντας με b λίτρα βενζίνη:

$$d(u,b) = \min_{\substack{j \neq u\ \kappa\alpha\iota \\ p(u)-p(j) \leq B}} \begin{cases} d(j,0) + \big(p(u) - p(j)\big) * c(u), & \alpha\nu\ c(j) < c(u)\ \kappa\alpha\iota\ b \leq p(u) - p(j) \\ d(j, B-b) + (B-b) * c(u), & \alpha\nu\ c(j) \geq c(u) \end{cases}$$

Ο λόγος για τον οποίο δεν εξετάζουμε φθηνότερη κορυφή στην οποία μπορούμε να φτάσουμε με τη βενζίνη που ήδη έχουμε είναι πως, είναι πάντοτε καλύτερο να ανεφοδιαστούμε κατευθείαν από τη φθηνότερη προσβάσιμη κορυφή.
Η λύση του προβλήματος είναι η τιμή $d(t, 0)$ και οι αρχικές συνθήκες είναι $d(0,0) = 0$, $d(0, b) = d(t, b) = \infty, \alpha\nu\ b > 0$.
Η συνολική πολυπλοκότητα του αλγορίθμου προκύπτει από $O(|V|)$ για τον υπολογισμό κάθε instance $d(u, b)$, για το πολύ $|V| * B$ instances. Συνολικά λοιπόν έχουμε $O(|V|^2 B)$.

Άσκηση 4:

Για να λύσουμε το πρόβλημα χρειάζεται να αντιστοιχίσουμε κάθε στρατιώτη και βάση εξτρεμιστών στον κοντινότερο στρατιώτη από τις δυνάμεις στρατού, ώστε να εξασφαλίσουμε πως θα επιτύχουμε τη βέλτιστη μεμονωμένη επίθεση κάθε φορά. Για να το κάνουμε αυτό θα χρησιμοποιήσουμε δυαδική αναζήτηση στις θέσεις των δυνάμεων στρατού a_i, αφού πρώτα τις ταξινομήσουμε σε $O(nlogn)$. Η αναζήτηση έχει πολυπλοκότητα $O\big((m+k)\log n\big)$ και έστω S ο πίνακας αντιστοιχίσεων που προκύπτει.

Κατόπιν, χρειάζεται να υπολογίσουμε ποιους εξτρεμιστές καλύπτει κάθε βάση. Για να το πετύχουμε αυτό ταξινομούμε τους εξτρεμιστές σε $O(mlogm)$ και κάνουμε δυαδική αναζήτηση στον ταξινομημένο πίνακά τους στις τιμές $\{b_i - d, b_i + d\}$, ώστε να βρούμε το εύρος κάλυψης κάθε βάσης με πολυπλοκότητα $O(klogm)$. Στη συνέχεια, κατασκευάζουμε έναν νέο πίνακα Ε' μήκους k στον οποίο τοποθετούμε στη θέση i του πρώτου στρατιώτη που καλύπτει η βάση j τον αριθμό 2^{j-1} και στη θέση i' του τελευταίου στρατιώτη που καλύπτει η βάση j τον αριθμό -2^{j-1}. Διατρέχουμε τον νέο πίνακα και υπολογίζουμε τα prefix sums του. Άρα, σε κάθε θέση του πίνακα μπορούμε εντοπίζοντας τους άσσους στη δυαδική του αναπαράσταση να βρούμε από ποιες βάσεις καλύπτεται.

Τέλος, ανάγουμε το πρόβλημα στο Minimum Cut ως εξής:

- Κατασκευάζουμε έναν γράφο G, με m κορυφές μία για κάθε εξτρεμιστή, k κορυφές μία για κάθε βάση και 2 κορυφές s, t.
- Συνδέουμε την κορυφή s με κάθε εξτρεμιστή που μπορούμε να αιχμαλωτίσουμε με μία ακμή $u = s \to e_i$ με χωρητικότητα $c(u) =$ minimum κόστος να κάνουμε επίθεση στον εξτρεμιστή, λαμβάνοντας υπόψη τον πίνακα S.
- Συνδέουμε επίσης την κορυφή t με κάθε βάση που μπορούμε να καταστρέψουμε με ακμή $u = b_i \to t$ με χωρητικότητα $c(u) =$ minimum κόστος να καταστρέψουμε τη βάση, λαμβάνοντας υπόψη τον πίνακα S.
- Συνδέουμε κάθε στρατιώτη e με κάθε βάση b που τον καλύπτει με ακμή $u = e \to b$ άπειρης χωρητικότητας.

Η λύση του παραπάνω στιγμιότυπου του Minimum Cut είναι και η ισοδύναμη λύση του προβλήματος. Συγκεκριμένα, όσες βάσεις βρεθούν μετά την τομή στην ίδια συνιστώσα με τον κόμβο t θα χρειαστεί να τις καταστρέψουμε κατά τη γενικευμένη επίθεση ενώ όσοι εξτρεμιστές βρεθούν στην ίδια συνιστώσα με τον κόμβο s θα χρειαστεί να αιχμαλωτιστούν.

Εξηγούμε την παραπάνω πρόταση:

- Εφόσον ζητάμε την ελάχιστη τομή αναζητούμε ακμές για τις οποίες η χωρητικότητα να ταυτίζεται με τη ροή, άρα οι ακμές άπειρης χωρητικότητας δεν θα συμμετέχουν.
- Αν βρούμε τιμή ροής για την οποία να δημιουργείται τομή στον γράφο, τότε κάθε βάση και οι στρατιώτες τους οποίους καλύπτει θα συνδέονται είτε με τον κόμβο s είτε με τον κόμβο t, δηλαδή δεν θα υπάρχει περίπτωση στο συνολικό άθροισμα μίας τομής να συμπεριλάβουμε και το κόστος καταστροφής μίας βάσης και το κόστος αιχμαλωσίας των εξτρεμιστών της.
- Άρα, βρίσκουμε τον συνδυασμό ακμών μικρότερου αθροίσματος που είτε θα αιχμαλωτίσουν τους εξτρεμιστές είτε θα καταστρέψουν τις αντίστοιχες βάσεις τους, χωρίς ποτέ να κάνουν και τις δύο ενέργειες και χωρίς να αφήνουν κανέναν εξτρεμιστή ελεύθερο. Επομένως, το πρόβλημα λύθηκε.

Η πολυπλοκότητα του minimum cut είναι $O(|E_G| * maximum_{flow})$, όπου οι τιμές για τις ακμές του G είναι $E_G = m * k + k$, στην χειρότερη περίπτωση που έχουμε συνεχείς επικαλύψεις βάσεων και maximum_flow = $\max\begin{bmatrix}\sum S_i, i \in εξτρεμιστές \\ \sum S_i, i \in βάσεις\end{bmatrix}$.