Εργασία 2

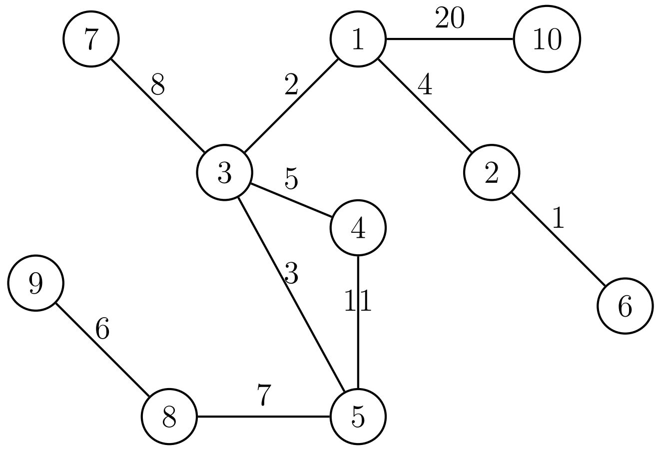
Απαντήσεις

Ονοματεπώνυμο: Παύλος Τομάζος **Α.Μ.:** 1115202200188
Ονοματεπώνυμο: Ευάγγελος Αργυρόπουλος **Α.Μ.:** 1115202200010

Θέμα 1

1. **ΑΛΗΘΗΣ:** Η ορθότητα του ερωτήματος έχει αποδειχθεί και στην σελ 74 των διαφανειών L11 του μαθήματος και είναι άμεσο αποτέλεσμα της ιδιότητας αποκοπής.

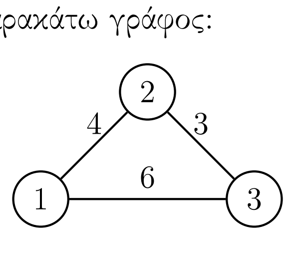
2. **ΨΕΥΔΗΣ:** Στην περίπτωση που η ακμή e με το μεγαλύτερο βάρος δεν είναι ακμή κύκλου, αλλά ενώνει άλλους κόμβους του γράφου, είναι αναγκαία για το ελάχιστο επικαλύπτον δέντρο. Για παράδειγμα ο παρακάτω γράφος:



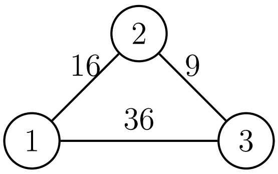
Στο παράδειγμα αυτό, η ακμή με το μεγαλύτερο βάρος είναι αυτή που ενώνει τους κόμβους 1 και 10, όμως αυτή δεν μπορεί να παραληφθεί από ένα ελάχιστο επικαλύπτον δέντρο γιατί θα χάσουμε κόμβους.

3. **ΑΛΗΘΗΣ:** Εφόσον τα κόστη c_e των ακμών είναι όλα θετικά, τότε μπορούμε να τα διατάξουμε σε μία σειρά όπου χωρίς βλάβη της γενικότητας θα ισχύει, $0 < c_{e_1} < c_{e_2} < c_{e_3} < \ldots$. Επειδή τώρα η συνάρτηση $f(n) = n^2$ είναι γνησίως αύξουσα για θετικές τιμές του n, θα ισχύει ότι $0 < (c_{e_1})^2 < (c_{e_2})^2 < (c_{e_3})^2 < \ldots$. Έτσι, τα κόστη στον γράφο G' θα έχουν την ίδια διάταξη με αυτή των κόστων στον αρχικό γράφο G και η δομή του γράφου θα παραμείνει η ίδια. Επομένως το ελάχιστο επικαλύπτον δέντρο T του αρχικού γράφου G θα είναι και πάλι ελάχιστο επικαλύπτον δέντρο για το γράφημα G'.

4. **ΨΕΥΔΗΣ:** Για παράδειγμα ο παρακάτω γράφος:



Στον γράφο αυτόν, η συντομότερη διαδρομή να πάμε από τον κόμβο 1 στον κόμβο 3 είναι ακολουθώντας την ακμή με βάρος 6. Όμως, αν τετραγωνίσουμε τα βάρη, θα έχουμε τον γράφο:



οπότε μπορούμε εύκολα να συμπεράνουμε ότι η διαδρομή της ακμής με βάρος 36 δεν είναι η συντομότερη, καθώς $16 + 9 = 25 < 36$.

Θέμα 2

1. Για να βρούμε το ελάχιστο πλήθος διαδρομών που πρέπει να κάνουμε ώστε να γεμίσουμε την δεξαμενή μπορούμε να χρησιμοποιήσουμε τον παρακάτω αλγόριθμο, ο οποίος με απληστία επιλέγει κάθε φορά τον κουβά με το μεγαλύτερο μέγεθος, δεδομένου ότι έχουμε τρεις κουβάδες. Για κάθε μέγεθος κουβά που επιλέγουμε, αφαιρούμε από το Μ τον αριθμό των λίτρων που γεμίζουμε την δεξαμενή με τους συγκεκριμένους κουβάδες που θα επιλέξουμε (π.χ. εάν επιλέξουμε 7 κουβάδες των 10 κιλών με $\mathrm{M} = 78$, τότε θα κάνουμε $\mathrm{M} = \mathrm{M} \mod 10 = 8$).

Algorithm 1 Find Bucket Routes

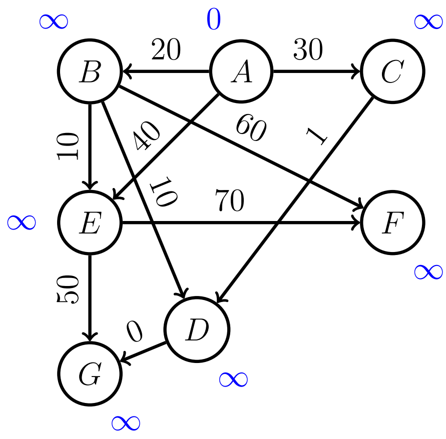
procedure FIND-BUCKET-ROUTES(M)
    $total\_routes \leftarrow 0$
    if $M \geq 10$ **then**
        $total\_routes \leftarrow total\_routes + \lfloor M/10 \rfloor$ ▷ Use as many 10-liter buckets as possible
        $M \leftarrow M \bmod 10$ ▷ Find the remaining liters
    end if
    if $M \geq 5$ **then**
        $total\_routes \leftarrow total\_routes + \lfloor M/5 \rfloor$ ▷ Use 5-liter buckets for the remaining liters
        $M \leftarrow M \bmod 5$
    end if
    $total\_routes \leftarrow total\_routes + M$ ▷ Use 1-liter buckets for any remaining liters
    return $total\_routes$
end procedure

2. Ο αλγόριθμος που παρουσιάσαμε δεν λειτουργεί στην περίπτωση που οι κουβάδες είναι 1, 10, 25 κιλών. Για παράδειγμα, αν $\mathrm{M} = 30$, με τον παραπάνω άπλιστο αλγόριθμο, θα πάρουμε τον κουβά των 25 κιλών και μετά θα έχουμε 5 λίτρα που απομένουν στοην δεξαμενή. Έτσι θα πάρουμε άλλους 5 κουβάδες του ενός κιλού. Συνολικά 6 διαδρομές. Το συμπέρασμα αυτό δεν είναι σωστό όμως, γιατί μπορούμε απλά να κάνουμε 3 διαδρομές με τον κουβά των 10 κιλών για να γεμίσουμε την δεξαμενή. Ο λόγος που δεν λειτουργεί ο αλγόριθμος, είναι γιατί το 25 δεν είναι πολλαπλάσιο του 10.

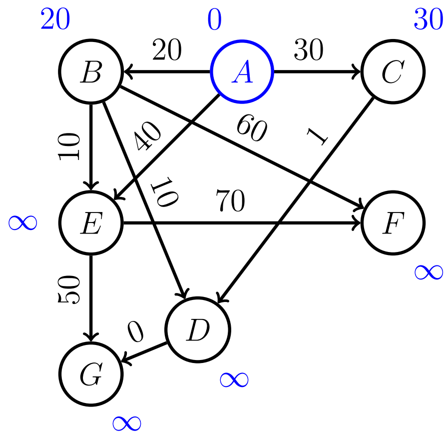
Θέμα 3

(α) Για τους σκοπούς της άσκησης θα χρησιμοποιήσουμε τον αριθμό μητρώου 1115202200010, επομένως $\alpha = 0, b = 1, c = 0$. Τότε $w_1 = 10\alpha + b = 1$, $w_2 = 10b + c = 10$ και $w_3 = 10c + \alpha = 0$.

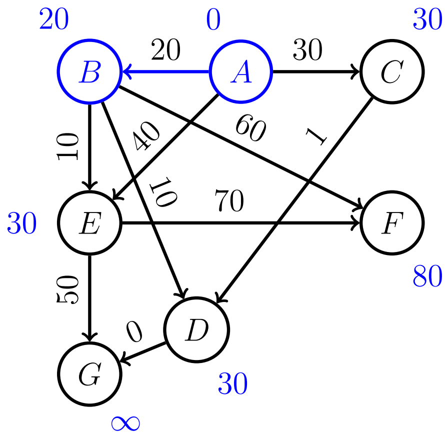
Οπότε, εφαρμόζωντας τον αλγόριθμο του Dijkstra έχουμε:



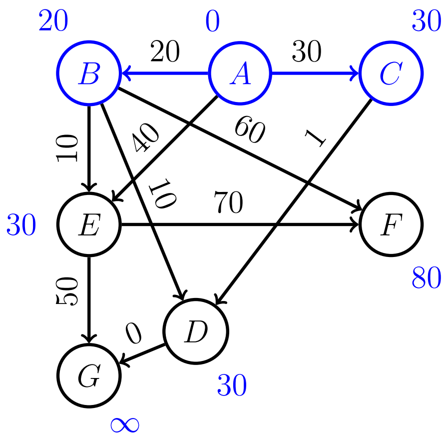
$1 : \{A, B, C, D, E, F, G\}$



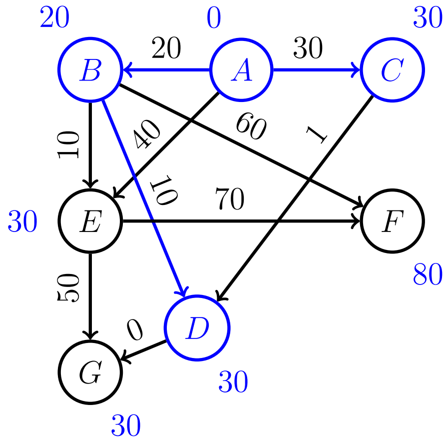
$2 : \{B, C, D, E, F, G\}$



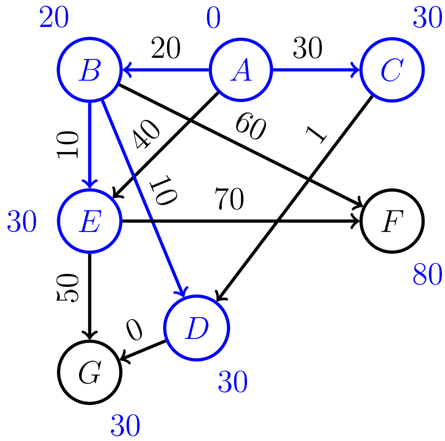
$3 : \{C, D, E, F, G\}$



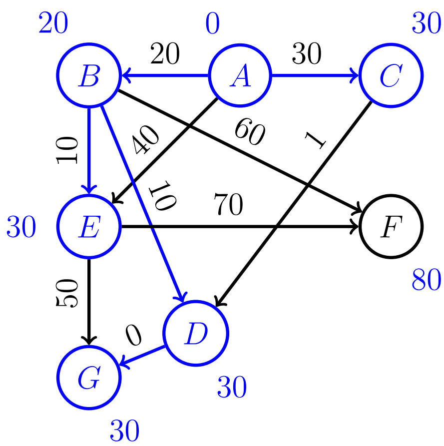
$4 : \{D, E, F, G\}$



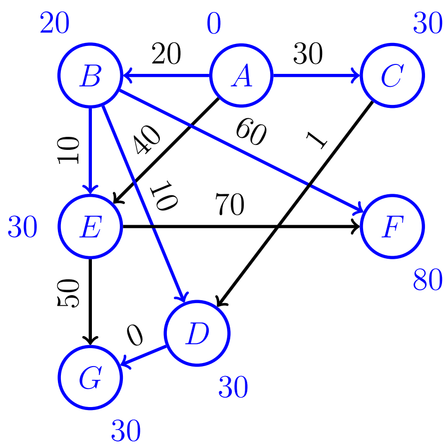
$5 : \{E, F, G\}$



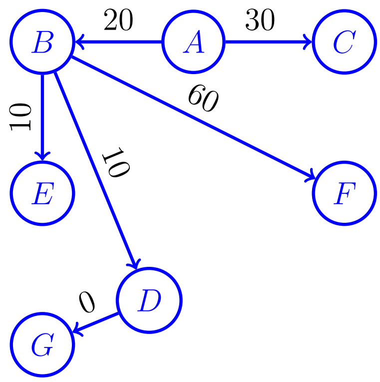
$6 : \{F, G\}$



$7 : \{F\}$

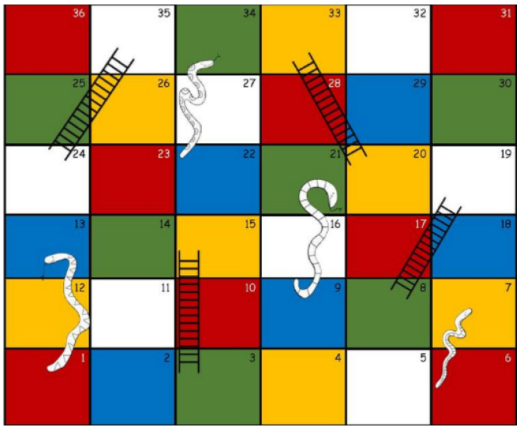


$8 : \emptyset$



Mininum Spanning Tree

(β) Το ταμπλό του παιχνιδιού Φιδάκι μπορεί να αναπαρασταθεί ως ένας γράφος, όπου κάθε κορυφή του θα απεικονίζει ένα κελί στο ταμπλό. Έτσι, το πρόβλημα της εύρεσης ελάχιστου πλήθους ζαριών για την μετάβαση από το αρχικό κουτάκι στο τελικό, απλοποιείται στο να βρούμε τον συντομότερο μονοπάτι στον γράφο. Κάθε κορυφή του γράφου έχει μία ακμή που οδηγεί στις επόμενες έξι κορυφές του γράφου, εφόσον αυτές δεν έχουν ούτε φίδι ούτε σκάλα. Αν υπάρχει φίδι ή σκάλα σε κάποια από τις έξι αυτές κορυφές, η ακμή από την τρέχουσα κορυφή οδηγεί απευθείας



Επιτραπέζιο Φιδάκι

στην αρχή της ουράς του φιδιού ή στην κορυφή της σκάλας. Καθώς τώρα, όλες οι ακμές έχουν ίδιο βάρος, μπορούμε να βρούμε εύκολα το συντομότερο μονοπάτι εφαρμόζωντας τον αλγόριθμο αναζήτησης κατά πλάτος (Breadth-First-Search). Έχουμε παραδώσει αντίστοιχο python script.

Algorithm 2 Find the minimum dice throws

procedure BFS-MIN-DICE-THROWS(G, end)
    $dist_v \leftarrow \infty$, for all vertices v in G
    $dist_1 \leftarrow 0$
    $queue \leftarrow \emptyset$
    $enqueue(queue, 1)$
    while $queue \neq \emptyset$ **do**
        $u \leftarrow dequeue(queue)$
        if u is end **then**
            return $dist_u$
        end if
        for each neighbor k in G **do**
            if $dist_k$ is ∞ **then**
                $dist_k \leftarrow dist_u + 1$
                $enqueue(queue, k)$
            end if
        end for
    end while
    return ∞
end procedure

Θέμα 4

(α) Άπλιστος Αλγόριθμος για την επίλυση του προβλήματος

Για την εύρεση του ελάχιστου απαιτούμενου αριθμού κεραίων που πρέπει να τοποθετήσουμε ώστε να καλύπτονται όλες οι πόλεις, μπορούμε να θεωρήσουμε ότι $0 < D[1] < D[2] < ... < D[n]$. Επίσης, υποθέτουμε ότι υπάρχει πάντα μία τουλάχιστον πόλη, δηλαδή $n \geq 1$. Συνεπώς, παραθέτουμε τον παρακάτω άπληστο αλγόριθμο που λύνει το πρόβλημα:

Algorithm 3 Find the minimum antennas to cover all cities

procedure FIND-MIN-ANTENNAS(r, D)
    $antenna\_count \leftarrow 1$
    $antenna\_range \leftarrow 2r$
    for $i \leftarrow 1$ **to** n **with step 1 do**
        if $D_i > antenna\_range$ **then**
            $antenna\_count \leftarrow antenna\_count + 1$
            $antenna\_range \leftarrow D_i + 2r$
        end if
    end for
    return $antenna\_count$
end procedure

Ο αλγόριθμος που παραθέσαμε είναι $O(n)$ και λειτουργεί άπληστα τοποθετώντας κάθε φορά μια κεραία σε απόσταση r από την κάθε πόλη η οποία δεν καλύπτεται από κάποια άλλη κεραία. Πιο συγκεκριμένα, αρχικά τοποθετούμε την πρώτη κεραία, σε απόσταση r από την πρώτη πόλη (έτσι ώστε οριακά να καλύπτεται), και προχωράμε και άλλη r απόσταση (συνολικά $2r$) αγνοώντας όλες τις πόλεις που βρίσκονται σε αυτήν την εμβέλεια (καθώς καλύπτονται από την κεραία που τοποθετήσαμε). Έπειτα, αναζητούμε την επόμενη πόλη που δεν καλύπτεται (δηλαδή $D_i > antenna\_range$) και επαναλαμβάνουμε την ίδια διαδικασία, δηλαδή τοποθετούμε κι άλλη κεραία σε απόσταση r (δεξιά) από αυτήν την πόλη, και ενημερώνουμε την εμβέλεια της καινούργιας κεραίας κατάλληλα.

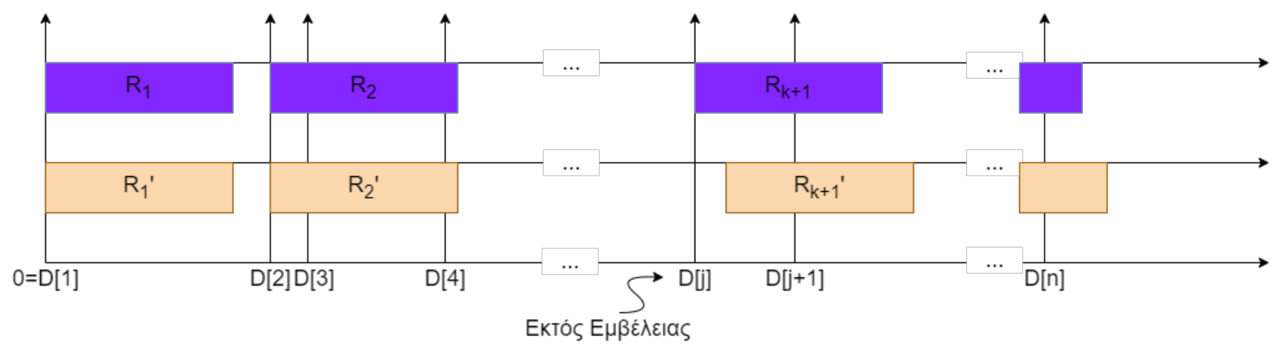
(β) Απόδειξη Ορθότητας Άπληστου Αλγορίθμου

Έστω $\{R_i\}_{i=1}^{m}$ η ακολουθία τοποθέτησης των m κεραιών που μετράει ο άπληστος αλγόριθμος με εμβέλεια το διάστημα $[R_i - r, R_i + r]$. Πιο συγκεκριμένα, γνωρίζοντας την άπληστη συμπεριφορά του αλγορίθμου, το εύρος κάθε κεραίας μπορεί να γραφτεί ως $[D[i], D[i] + 2r]$.

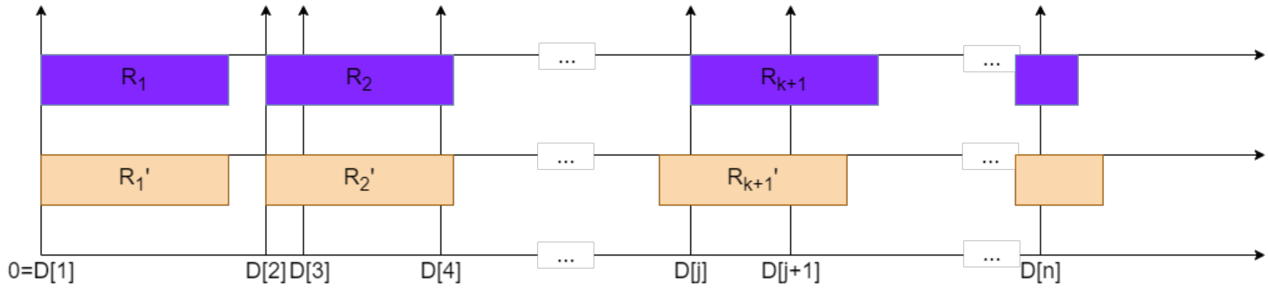
Θα αποδείξουμε με απαγωγή σε άτοπο ότι η λύση του άπληστου αλγορίθμου είναι βέλτιση.

Έστω μία βέλτιση ακολουθία τοποθέτησης κεραιών $\{R'_i\}_{i=1}^{l}$ με τους k πρώτους όρους της να είναι ίδιοι με της ακολουθίας $\{R_i\}_{i=1}^{m}$, ώστε το k να είναι το μέγιστο δυνατό. Η πρώτη διαφορά, λοιπόν, τοποθέτησης θα γίνεται στην $(k+1)-$οστή κεραία.

Εάν η κεραία $k+1$ στην βέλτιστη λύση βρίσκεται πιο δεξιά από την τοποθέτηση που έχει θεωρήσει ο άπληστος αλγόριθμος, τότε η πόλη j που είναι στο αριστερό σύνορο της εμβέλειας της κεραίας R_{k+1}, πλέον δεν θα καλύπτεται από καμία κεραία της ακολουθίας $\{R'_i\}_{i=1}^{l}$. Αυτό σημαίνει ότι η ακολουθία $\{R'_i\}_{i=1}^{l}$ δεν αποτελεί λύση, διότι θα υπάρχει πόλη εκτός της εμβέλειας όλων των κεραιών. Άτοπο.



Επομένως, η κεραία $k+1$ στην βέλτιστη λύση, θα βρίσκεται πιο αριστερά από ότι στην λύση που προκύπτει από τον άπληστο αλγόριθμο. Αυτό σημαίνει πως η εμβέλεια της κεραίας R'_{k+1} θα ξεκινάει πριν από την πόλη j. Επειδή όμως όλες οι προηγούμενες κεραίες είναι όμοιες και στις δύο λύσεις, σημαίνει πως οι πόλεις πριν από την j έχουν ήδη καλυφθεί. Επομένως, το διάστημα εμβέλειας της R'_{k+1} που δεν ανήκει στην εμβέλεια της R_{k+1} $[R'_{k+1} - r, D[j])$ δεν έχει κάποιο αντίκτυπο στη λύση. Έτσι, μπορούμε να κατασκευάσουμε μία επίσης βέλτιστη λύση αν απλά αντικαταστήσουμε την κεραία R'_{k+1} με την R_{k+1}.



Καταλήξαμε σε άτοπο, διότι υπάρχει βέλτιστη λύση με περισσότερους από k όμοιους όρους, ενώ υποθέσαμε ότι το k είναι ο μέγιστος αριθμός ομοιοτήτων (των πρώτων όρων). Συνεπώς, η λύση του άπληστου αλγόριθμου που παραθέσαμε είναι βέλτιστη.

Θέμα 5

(1) Αντιπαραδείγματα στις δοθείσες στρατηγικές

Σε όλα τα αντιπαραδείγματα θεωρούμε ότι υπάρχουν μόνο δύο παιδιά, δηλαδή $n = 2$ με εκτιμώμενους χρόνους κολύμβησης s_1, s_2 και ποδηλασίας b_1, b_2 αντίστοιχα.

- Υποθέτουμε ότι $s_1 = 5$, $s_2 = 7$, $b_1 = 15$, $b_2 = 5$. Αν τοποθετήσουμε τα παιδιά σε φθίνουσα σειρά βάσει του χρόνου κολύμβησής τους, το παιδί 2 ξεκινά πρώτα και τελειώνει σε $s_2 + b_2 = 12$ λεπτά, ενώ το παιδί 1 θα χρειαστεί από την έναρξη του διαγωνισμού συνολικά $s_2 + s_1 + b_1 = 7 + 5 + 15 = 27$ λεπτά (προσθέτωντας και τον χρόνο που θα περιμένει το άλλο παιδί στην πισίνα). Επομένως, η διοργανωση θα λήξει σε 27 λεπτά. Η λύση αυτή όμως δεν είναι βέλτιση διότι αν ξεκινήσει πρώτα το παιδί 1, θα χρειαστεί χρόνο $s_1 + b_1 = 5 + 15 = 20$ λεπτά ενώ το παιδί 2 θα χρειαστεί $s_1 + s_2 + b_2 = 5 + 7 + 5 = 17$. Άρα, η διοργάνωση θα διαρκήσει, συνολικά, 20 λεπτά.

- Θεωρούμε τους εκτιμώμενους χρόνους $s_1 = 1$, $s_2 = 2$, $b_1 = 15$, $b_2 = 5$. Τότε $s_1 + b_1 = 16$ λεπτά και $s_2 + b_2 = 7$ λεπτά. Αν διατάξουμε τα παιδια σε φθίνουσα σειρά βάσει του αθροίσματος των χρόνων κολύμβησης και ποδηλασίας, θα ξεκινήσει πρώτα το παιδί 2 και θα χρειαστεί 7 λεπτά ενώ το παιδί 1 θα χρειαστεί $2 + 16 = 18$ λεπτά από την αρχή της διοργάνωσης. Οπότε ο συνολικός χρόνος της διοργάνωσης είναι 18 λεπτά. Αν όμως, ξεκινήσει πρώτα το παιδί 1, θα χρειαστεί 16 λεπτά ενώ το παιδί 2, $1 + 7 = 8$, με συνολικό χρόνο διοργάνωσης 16 λεπτά.

- Έστω ότι οι χρόνοι των δύο παιδιών είναι $s_1 = 1$, $b_1 = 1$, $s_2 = 2$, $b_2 = 2$. Ταξινομώντας την σειρά εμφάνισης των παιδιών σε αύξουσα σειρά βάσει του χρόνου ποδηλασίας τους, θα ξεκινήσει πρώτα το παιδί 1, το οποίο θα χρειαστεί $s_1 + b_1 = 1 + 1 = 2$ λεπτά. Ύστερα, θα ακολουθήσει το παιδί 2 με χρόνο (από την έναρξη της διοργάνωσης) $s_1 + s_2 + b_2 = 1 + 2 + 2 = 5$ λεπτά. Άρα, η συνολική διάρκεια θα είναι είναι 5 λεπτά. Η στρατηγική αυτή, όμως, δεν οδηγεί σε βέλτιση λύση. Αν ξεκινήσει το παιδί 2, ο χρόνος που θα χρειαστεί είναι $s_2+b_2 = 2+2 = 4$ λεπτά ενώ για το παιδί 1 θα είναι $s_2 + s_1 + b_1 = 1 + 1 + 2 = 4$ λεπτά. Συνολικά 4 λεπτά.

(2) Άπληστος Αλγόριθμος για την επίλυση του προβλήματος - Απόδειξη ορθότητας

Για την εύρεση του νωρίτερου δυνατού χρόνου λήξης της διοργάνωσης μπορούμε να χρησιμοποιήσουμε έναν άπληστο αλγόριθμο που ταξινομεί τα παιδιά με βάση τον εκτιμώμενο χρόνο ποδηλασίας τους b_i σε φθίνουσα σειρά και έπειτα τα επιλέγει με την διάταξη που προκύπτει.

Θα αποδείξουμε ότι ο άπληστος αυτός αλγόριθμος είναι βέλτιστος. Έστω μία οποιαδήποτε βέλτιστη λύση η οποία δεν χρησιμοποιεί την προταθείσα διάταξη. Τότε, αυτή η λύση θα έχει τουλάχιστον δύο παιδιά, i και j, τέτοια ώστε το j να ξεκινάει αμέσως μετά το i με $b_i < b_j$. Παρατηρούμε ότι σε αυτήν την βέλτιστη λύση το παιδί j που ξεκινάει μετά το i, θα κάνει περισσότερη ώρα ποδήλατο και έτσι θα τερματίσει τελευταίο ανάμεσα στα δύο αυτά παιδιά σε χρόνο $(\sum s_k) + s_i + s_j + b_j$ από την αρχή της διοργάνωσης (όπου $\sum s_k$ ο χρόνος κολύμβησης των προηγούμενων παιδιών). Από αυτό μπορούμε να συμπεράνουμε πως αν αντιστρέψουμε το i με το j:

- το παιδί j θα ξεκινήσει πιο νωρίς, άρα θα τελειώσει και πιο νωρίς από πριν.
- το παιδί i που θα ξεκινήσει μετά το j, θα κάνει λιγότερη ώρα ποδήλατο από το j, και θα τελειώσει και αυτό νωρίτερα από ότι θα τελείωνε το παιδί j πριν την αντιστροφή. Αυτό συμβαίνει επειδή το παιδί i μετά την αντιστροφή θα χρειαστεί $(\sum s_k) + s_j + s_i + b_i$ και έχουμε $b_i < b_j \Rightarrow (\sum s_k) + s_j + s_i + b_i < (\sum s_k) + s_i + s_j + b_j \Rightarrow$ Χρόνος του i μετά $<$ Χρόνος του j πριν.

Οι δύο παραπάνω παρατηρήσεις συνεπάγονται ότι μετά την αντιστροφή δεν θα υπάρξει αύξηση στον συνολικό χρόνο. Με άλλα λόγια, μπορούμε απλά να αντιστρέψουμε τον συνδυασμό (i, j) της βέλτιστης λύσης, και να παραμείνει βέλτιστη. Την διαδικασία αυτή μπορούμε να την επαναλάβουμε απαλείφοντας όλες τις αντιστροφές, χωρίς αύξηση του χρόνου ολοκλήρωσης της διοργάνωσης. Όταν εξαλειφθούν όλες οι αντιστροφές, η τελική βέλτιστη λύση θα ταυτίζεται με αυτήν που προκύπτει από τον άπληστο αλγόριθμο, αποδεικνύοντας ότι είναι και εκείνη βέλτιστη.