

1^η Σειρά Προγραμματιστικών
ΕΠΙΘΕΤΟ: ΣΕΒΑΣΤΟΥ
ΟΝΟΜΑ: ΝΙΚΟΛΕΤΑ
Α.Μ.: 711 514 22 00015
ΜΑΘΗΜΑ: Αλγόριθμοι (ΑΛΜΑ)

2023-24

Άσκηση 1 Προσπαθώντας να ερμηνεύσουμε

PREFIX (A, median, N)

% $N = A.length$

PR = new-matrix (1, N) % νέος πίνακας $1 \times N$

% εἴσεν του prefix sum πίνακα

for $i=1$ to N

if $A[i] \geq median$

PR[i] = 1

else

PR[i] = -1

end

if $i > 1$

PR[i] = PR[i] - PR[i-1]

end

end

return PR

CLOSE-TO-MEDIAN (med, k, N, A)

PRE = PREFIX (A, med, N)

% εἴσεν εἰς τον median

mini = 0

maxi = PRE[N]

for $i=k$ to N

mini = min (mini, PRE[i-k+1])

maxi = max (maxi, PRE[i] - mini)

end

if maxi > 0

return True % εἴσος

else

return False % εἴσος

end

% Τα στοιχεία $i-k+1, \dots, i$ είναι k το μέγεθος

MAX-MEDIAN (N, k, A)

% A: o nivakos unv C_i

$l=1$, $maxi=-2$

$r=N$, $mid = \lfloor \frac{l+r}{2} \rfloor$

while $l \leq r$

if (CLOSE-TO-MEDIAN (mid, k, N, A))

$l=l+1$ % fenorè ious kai figadùttero

$maxi=mid$

else

$r=r-1$ % xeràjofean mikròttero

end

$mid = \lfloor \frac{l+r}{2} \rfloor$

end

return maxi

Άσκηση 2

mergesort (A, l, r)

if $l < r$

mid = $\lfloor \frac{l+r}{2} \rfloor$

return merge (mergesort (A, l, mid), mergesort (A, mid+1, r))

else

return A

end

merge (A, B)

% A, B είναι δύο ταξινομημένοι πίνακες σε αύξουσα

% σειρά με βάση την m σειρά τους

m, k = A.length

n, l = B.length s = max {m, n}

% Σημειώστε πίνακα για αποθήκευση της "ένωσης" των

% A, B

C = new-matrix (s, k+l) % s x (k+l) πίνακας

i = 1, j = 1, r = 1

while (i ≤ k & j ≤ l)

if A[i, i] < B[j, j]

C[:, r] ← A[:, i] % A[i, i] είναι η 1^η στήλη

% του A

i = i + 1

else

C[:, r] ← B[:, j]

j = j + 1

end

r = r + 1

end

while i ≤ k

C[:, r] ← A[:, i]

i ← i + 1, r ← r + 1

end


```

while j ≤ l
    C[i, r] ← B[:, j]
    j ← j+1, r ← r+1
end

```

```

return C

```

% τέλος merge

cost (p, w, b)

m = p.length *% τα p, w θα έχουν ίδιο μήκος*

C = new-matrix (1, m) *% πίνακας 1x m*

```

for i = 1 to m

```

C[i] = b * w[i] - p[i] *% αριθμοί από*

*% p[i] - b * w[i] για να βγάζει ο Kruskal*

% μετά max span. tree

```

end

```

```

return C

```

KRUSKAL (E, m, n, ..., c)

% είπεν max. span. tree γιατί τα βάρη θα

% είναι αρνητικά

*% E: πίνακας ακμών 4x m όπου η γραμμή 3 αριθμεί
620*

% w: -// - βάρη

% και n 4ⁿ βζον p: πίνακας κόστους για ακμές

cost = (E(4,:), E(3,:), c)

E = mergesort (E, 1, m)

Q = new-queue

```

for i = 1 to m

```

makeSet (i) *% συνήλωση συνόλου που περιέχει*

% το i

i = 1, S_p = 0, S_w = 0

```

while i ≠ m then

```

i ≠ find (E[1, i]) ≠ find (E[2, i])

% find(v): εντοπίζει μοναδικά καθορισμένο

% αντιστάθμισμα του συνόλου όπου ανήκει το v


```

enqueue(set( $E[1,i]$ ,  $E[2,i]$ ),  $Q$ )
% set( $u,v$ ) : είναι σύνολο των  $u,v$ 
union( $E[1,i]$ ,  $E[2,i]$ )
 $S_p = S_p + E[4,i]$ 
 $S_w = S_w + E[3,i]$ 
end % union( $u,v$ ): ακολουθία τα σύνολα που
 $i = i+1$  % περιέχουν τα  $u,v$  με την ένωση τους
end % και δίνει ένα αντιστοιχισμο των συνόλων.
return  $Q$ ,  $S_p$ ,  $S_w$ 

```

<pre> max(p) % p: πίνακας $m = p[1]$ $k = p.length$ for $i = 2$ to m if $m < p[i]$ $m = p[i]$ end end return m </pre>	<pre> min(p) % p: πίνακας $m = p[1]$ $k = p.length$ for $i = 2$ to m if $m > p[i]$ $m = p[i]$ end end return m </pre>
---	---

```

gcd( $a,b$ ) %  $O(\log(a+b))$ 
 $i = a$ ,  $j = b$ 
while ( $i > 0$ ) and ( $j > 0$ )
    if  $i > j$ 
         $i = (i \bmod j)$ 
    else
         $j = (j \bmod i)$ 
    end
end
return ( $i+j$ )

```


BIN-S-MST (n, m, E)

% n : αριθμός κορυφών

% m : αριθμός ακμών

% E : πίνακας $4 \times m$ όπου η 3^η στήλη είναι

% τα w των ακμών και η 4^η τα p

$p = E[4, :]$, $w = E[3, :]$

$p_{\max} = \max(E[4, :])$, $p_{\min} = \min(E[4, :])$

$w_{\max} = \max(E[3, :])$, $w_{\min} = \min(E[3, :])$

$b = p_{\max}/w_{\min}$

$a = p_{\min}/w_{\max}$

$c = (a+b)/2$

$\text{cost} = \text{cost}(p, w, c)$

$H, S-p, S-w = \text{Kruskal}(E, m, n, c)$

while $b-a \geq 10^{-8}$

$d = S-p - c * S-w$

if $(d > 0)$

$a = c$

elif $(d < 0)$

$b = c$

elif $(d == 0)$

return $S-p / \gcd(S-p, S-w),$
 $S-w / \gcd(S-p, S-w)$

end

$c = (a+b)/2$

$H, S-p, S-w = \text{Kruskal}(E, m, n, c)$

end

return $S-p / \gcd(S-p, S-w), S-w / \gcd(S-p, S-w)$