AISD lista 7

Dominik Szczepaniak

May 20, 2024

1 Zadanie 1

Ponieważ w naszej operacji Union(A, B) przypisujemy A jako ojca B, to mamy O(1). Jeśli wszystkie Union są przed find to najpierw wykonamy wszystkie Uniony w złożoności O(1).

Teraz mamy same operacje Find(x). Załóżmy, że elementów jest n. Załóżmy, że wszystkie wierzchołki są na początku odznaczone. Zaznaczamy wierzchołek tylko gdy przejdziemy przez niego jakimś findem (również gdy szliśmy z kogoś niżej do góry). W takim razie widać, że jeżeli nie ma żadnych unionów po wykonaniu finda, to jeśli wierzchołek będzie zaznaczony raz to będzie od razu podpięty pod ojca zbioru. Czyli każdy wierzchołek możemy zaznaczyć co najwyżej raz każdym findem, czyli mamy złożoność co najwyżej O(n).

Union - O(n) + Find O(n) = O(2n) = O(n)

2 Zadanie 2

Możemy użyc drzewa Splay. Jeśli mamy insert to normalnie sobie insertujemy. Jeśli mamy min(i) to splayujemy pierwsza wartość większą niż i, i usuwamy lewe drzewo. Jak mamy deletemin to po prostu idziemy na maxa na lewo i usuwamy lewą wartość. Jeśli tutaj będziemy mieć długą ścieżkę w prawo to nam to nic nie wadzi, bo zawsze usuwamy minimalny element, więc usuniemy po prostu korzeń.

Problem - możemy usuwać i inserować ostatni element na zmiane i mamy ciągle złożoność O(n), więc łącznie $O(n^2)$

==========

Załóżmy, że ciąg instrukcji sigma = sigma1, E, sigma2, E, sigma3, E, itd..

W sigmie ściagamy wszystkie kolejne operacje insert jako jedną operację.

Czyli jeśli mielibyśmy Insert(3), Insert(2), Insert(4) to robimy to jako Insert(3, 2, 4).

E to operacje ExtractMIN

Tworzymy n zbiorów dla algorytmu Union Find tak żeby te zbiory zawierały kolejne sigmy1, sigma2, ...

Czyli tworzymy k takich zbiorów (k to liczba insertów), do których należą wszystkie elementy z $sigma_k$. Jeśli jest mniej niż k sigm to zbiory ostatnie są puste.

Robimy dwie tablice PRED i SUCC które słuzą do stworzenia podwójnie sklejanej linked sorted listy dla tych wartości j dla których $sigma_j$ istnieje.

Na starcie PRED[j] = j-1 dla 1 <= j <= k+1 oraz SUCC[j] = j+1 dla 0 <= j <= k.

for i in 1 to n:

j = find(i)

if j <= k:

print i usuniete przez j-ty $extract_min$

union(j, succ[j], succ[j])

succ[pred[j]] = succ[j]

pred[succ[j]] = pred[j]

czas działania tego algorytmu jest ograniczona przez UNION FIND, wiec taka jest złożoność

3 Zadanie 4

No nie. Załóżmy, że mamy ścieżkę długości n. Jeśli za każdym razem będziemy podwieszać do swojego dziadka. To ostatni wierzchołek idzie do n-2, n-2 do n-4, n-4 do n-6, ... Czyli zmniejszymy tylko długość drogi o dwukrotnie. Skoro zmniejszymy dwukrotnie, to po dwóch razach czterokrotnie itd. Czyli w log(n) krokach zmniejszymy o tyle ile zmniejszylibyśmy w 1 kroku. Czyli złożoność tutaj byłaby $log^2 n$.