NIO 2021 finale

Løsningsforslag

Tautrekking

Ønsker å finne største verdi K slik at tauet har enten “r”*K + “b”*K eller “b”*K + “r”*K som substreng.

Prøve alle verdier av K, og sjekke med string.find()

Binærsøke på K, og sjekke med string.find()

Ingen av delene ideelle. string.find() kan være treg hvis søkestrengen har mye repetisjoner.

Tautrekking

Bryt tauet ned til lengden på de ensfargede seksjonene.

tau = rrrbbrrrrbbbrr => lengder = [3, 2, 4, 3, 2]
svaret er da max (min(lengder[i], lengder[i +1]))

Iterer over strengen én gang. Man begynner en ny seksjon hvis fargen på denne taubiten er forskjellig fra den forrige. Hvis ikke legger man taubiten på forrige seksjon.

Trenger ikke å bygge opp v-arrayen. Trenger kun å huske forrige verdi.

Togtur

Ligner på en shortest path oppgave. For V = 0 er den standard shortest path.

Dette løses typisk med Djikstra.

Togtur - Djikstra

La d[i] bretagne avstand fra by 0 til by i. d[0] =0.

Initialiser en priority queue (min heap) med {avstand 0, by 0}

Så lenge køen ikke er tom:
Hent og fjern element (avstand, u) med lavest avstand i køen.
Hvis u ikke alllerede er prosessert, sett d[u] = avstand
For alle kanter (lengde, til) ut av u, legg (avstand+lengde, til) i køen
Marker u som prosessert

Togtur - gratiskanter

Konstruer V+1 “lag” av grafen.

Lag i representerer at du har i tilgjengelige billetter.

d[V][0] = 0

Hvert lag av grafen har de samme kantene som den opprinnelige grafen.

I tillegg gir alle toglinjer deg kanter fra hvert lag (untatt det nederse) til det neste laget (altså der hvor du har 1 billett færre) som har kostnad 0

Topografi

Med litt vurdering set man at største garanterte høydeforskjell er lik den lengste stien i den rettede grafen som snøballsporene lager.

Var også en “lengste sti”-oppgave i runde 2, men der på en annen type graf.

Litt mer utfordrende siden du her må konstruere grafen selv, men begrenset størrelse på rutenettet gjør at man kan bruke O(N*M) minne.

Glatte trær

Første observasjon:

Rekkefølge på swap operasjoner har ingenting å si. Hver node trenger kun å utføre swap maksimalt én gang.

Enkel løsning:

For hvert subset av noder, prøv ut hvordan ruheten blir hvis nøaktig disse nodene får en swap operasjon.

Tid: O(2^N * N). Løser subtask 1.

Glatte trær

Andre observasjon:

Hvert subtre lager en sammenhengende sekvens i traverseringen.

Hvert subtres bidrag til ruhet er dermed kun påvirket av hvilke tall som havner rett før og rett etter.

Gir oss en mulighet til å bruke dynamisk programmering.

Glatte trær

```
best(node, verdi_før, verdi_etter):
  if node == null:
    return 0 if verdi_før == null || verdi_etter == null else
                    abs(verdi_før, verdi_etter)
  else:
    return min(
        best(node.venste, verdi_før, node.verdi) +
        best(node.høyre, node.verdi, node.høyre),
        best(node.høyre, verdi_før, node.verdi) +
        best(node.venste, node.verdi, node.høyre))

print(best(0, null, null))
```

Glatte trær

Ved å rekursere ned til “null” noder, så slipper vi å håndtere forskjellige tilfeller av at noder har 0, 1 eller 2 barn for å se hvordan dette nodens verdi selv påvirker summen. Nå blir det i stedet betraktet når vi kommer ned til et “tomt tre” og venstre-verdi og høyre-verdi ender opp med å berøre hverandre.

Memoiser resultatet for forskjellige verdier av (node, verdi_før, verdi_etter) for å få en løsning som bruker O(N^3) tid og O(N*max_verdi^2) minne.

Glatte trær

Gruppe 3:
venstre_verdi og høyre_verdi må alltid være verdien til en node (eller null). Bruk dette for å redusere minne ned til O(N^3) også.

Gruppe 4:
Enten venstre_verdi eller høyre_verdi vil alltid være verdien av foreldrenoden. Siden hvilken verdi som er venstre_verdi og høyre_verdi egentlig ikke spiller noen rolle kan vi gjøre det slik at det alltid er høyre_verdi som endrer seg, og vi får O(N^2) tid/minne

Slange

Finn intervallet med lengde L hvor summen av de L-K minste elmentene er minst mulig.

For små verdier av N (og dermed både L og K) kan det løses ved å beregne for hvert eneste gyldige intervall.

For støtte verdier trenger man en bedre datastruktur.

Slange

Ønskede operasjoner:

A) Legge till et tall
B) Fjerne et tall
C) Beregne summen av alle tallene unntatt de K største

Multiset gjør operasjon A og B effektivt for oss, men ikke C.

C vil kreve at man itererer over size-K elementer og summerer selv.

Slange

Løsning:

Bruk to multiset: ett for de K største tallene og ett for resten

Hold selv styr på summen av “resten”

Slange

```
insert(x):
  stor.insert(x)
  if (size(stor) > K):
    flytt minste element fra stor til liten

delete(x):
  if liten.contains(x):
    liten.delete(x)
  else:
    stor.delete(x)
    if size(stor) < K and size(liten) > 0:
      flytt største element fra liten til stor

#TODO hold styr på summen av “liten” hver gang den endrer seg
```

Nordic Olympiad in Informatics

Åpen konkurranse for videregåendeskoleelever

24. 3. 2021 (Onsdag)

https://noi2021.nio.no

Resultater

10	193	Jonas Elvedal Hole
9	198	August Asp
8	201	Falk Elvedal Bruskeland
7	220	Mathias Harkestad Bæverfjord
6	238	Inge Grelland
5	272	Magnus Eide-Fredriksen
4	349	Christoffer Grøndal Tryggestad
3	400	Andreas Alberg
2	410	Mathias Presthagen
1	500	Magnus Hegdahl