APS 2 — tretja domača naloga

Rok za oddajo: nedelja, 22. marca 2026

Dinamično dvojiško iskanje

Ozadje

Dobro vemo, da je časovna zahtevnost iskanja posameznega elementa v urejeni tabeli z n elementi enaka $O(\log n)$, vstavljanje v takšno tabelo pa zahteva čas $O(n)$. Lahko obe operaciji izvedemo v času $o(n)$, ne da bi morali poseči po čem drevesnem?

Takole gre: namesto da bi elemente hranili v eni sami urejeni tabeli, jih hranimo v urejenih tabelah velikosti 1, 2, 4, 8, 16 itd. Vsaka od teh tabel je bodisi povsem polna bodisi povsem prazna. Odgovor na vprašanje, ali je določena tabela polna ali prazna, je enolično določen s številom elementov v celotni strukturi (n): če je na i-tem mestu (od zadaj naprej, začenši z 0) v dvojiškem zapisu števila n enica, bo tabela z velikostjo 2^i polna, če je tam ničla, pa bo omenjena tabela prazna. Na primer, če je $n = 13 = 1101_{(2)}$, bodo tabele z velikostmi $2^0 = 1$, $2^2 = 4$ in $2^3 = 8$ polne, tabela velikosti $2^1 = 2$ pa prazna.

Sedaj se nam že približno svita, kako izvedemo iskanje in dodajanje. Pri iskanju ni prav dosti dilem: z dvojiškim iskanjem preiščemo vsako polno tabelo posebej. Pri dodajanju pa ravnamo takole: poiščemo najkrajšo prazno tabelo (recimo, da je to tabela velikosti k), nato pa vse krajše tabele skupaj z novim elementom zlijemo v tabelo velikosti k, ki ima ravno dovolj prostora za vse prišleke. Krajše tabele se tako izpraznijo.

Oglejmo si primer. Recimo, da v strukturo po vrsti dodamo elemente 50, 20, 80, 30, 70, 40, 60 in 10:

$$[[]] \to [[50]] \to [[], [20, 50]] \to [[80], [20, 50]] \to [[], [], [20, 30, 50, 80]] \to$$
$$[[70], [], [20, 30, 50, 80]] \to [[], [40, 70], [20, 30, 50, 80]] \to [[60], [40, 70], [20, 30, 50, 80]] \to$$
$$[[], [], [], [10, 20, 30, 40, 50, 60, 70, 80]]$$

Naloga

Napišite program `tabelatabel.cpp`, ki po opisanem postopku vzdržuje opisano podatkovno strukturo. Program naj prebere število u (število ukazov) in zaporedje u ukazov, izpiše pa naj rezultate izvedbe teh ukazov.

Ukazi so sledeče oblike:

- `+` x: V strukturo dodaj število x.
- `?` x: Če je število x v strukturi prisotno, izpiši njegov indeks v naraščajoče urejenem zaporedju vseh števil, ki jih hrani struktura. Če število v strukturi ni prisotno, izpiši vrednost $(-i - 1)$, kjer je i indeks, na katerega bi se število x uvrstilo v omenjenem zaporedju. (Če je število x manjše od najmanjšega elementa strukture, je torej treba izpisati -1, če je večje od največjega elementa strukture, pa $-n - 1$, kjer je n število elementov strukture.)
- `i`: Po zgledu spodnjih primerov izpiši tabele, ki tvorijo strukturo.

Vhod

V prvi vrstici je podano število $u \in [1, 10^5]$, nato pa sledi u vrstic, od katerih vsaka vsebuje po en ukaz. Števila x, ki nastopajo kot argumenti ukaza `+`, pripadajo intervalu $[-10^9, 10^9]$ in so si med seboj različna (struktura tako nikoli ne bo vsebovala duplikatov). Argumenti ukaza `?` prav tako pripadajo intervalu $[-10^9, 10^9]$, ni pa nujno, da so si med seboj različni.

Vsak testni vhod je pripravljen tako, da skupna velikost pripadajočega izhoda ne presega 2 MiB.

V 50% testnih primerov nastopajo samo ukazi `+` in `i`.

Izhod

Izpišite toliko vrstic, kot je ukazov `?` in `i`. V vsaki vrstici izpišite rezultat izvedbe pripadajočega ukaza.

Testni primer 3

Vhod:

```
30
i
? 30
+ 50
i
+ 20
i
+ 80
i
+ 30
i
? 40
? 20
+ 70
i
+ 40
i
? 40
? 50
? 70
+ 60
i
? 65
+ 10
i
? 15
? 5
? 30
? 75
? 80
? 85
```

Izhod:

```
[]
-1
[[50]]
[[], [20, 50]]
[[80], [20, 50]]
[[], [], [20, 30, 50, 80]]
-3
0
[[70], [], [20, 30, 50, 80]]
[[], [40, 70], [20, 30, 50, 80]]
2
3
4
[[60], [40, 70], [20, 30, 50, 80]]
-6
[[], [], [], [10, 20, 30, 40, 50, 60, 70, 80]]
-2
-1
2
-8
7
-9
```

Oddaja naloge

Oddajte datoteko `tabelatabel.cpp`.

Neobvezni (a dejansko še pomembnejši) del

Programiranje je seveda pomembna veščina, z vidika algoritmike pa so naslednja vprašanja še pomembnejša:

- Do nadaljnjega se osredotočimo samo na operacijo dodajanja. Z agregatno analizo ocenite njeno amortizirano časovno zahtevnost.
- Kako bi nastavili računovodsko ceno operacije dodajanja, da bi za poljubno zaporedje n dodajanj veljalo $\sum_{i=1}^{n} \hat{c} \geqslant \sum_{i=1}^{n} c_i$?
- Določite potencial podatkovne strukture, tako da bo usklajen z računovodsko ceno.
- Koliko časa traja posamezna operacija iskanja v najslabšem primeru?
- Koliko časa traja zaporedje n parov operacij dodaj in poišči?
- Kako bi učinkovito realizirali operacijo brisanja? (Namig: kako rokovati z »luknjami«?)

Odgovorov na ta vprašanja ne oddajte, kljub temu pa jih (*za svojo korist!*) skrbno zapišite na dobri stari papir.