Timbre

Folosim o coadă de priorități în care elementele vor fi ordonate crescător pentru ca elementul din vârful cozii să fie mereu minimul. Întrucât în coadă vom stoca costurile timbrelor și ni se cere costul minim.

Citim de la tastatură perechile margine-superioară și costul timbrului și le reținem într-o structură de date de tip vector de perechi. Apoi acest vector îl ordonăm crescător după margine superioară pentru ca să îl parcurgem de la sfârșit pentru că o să începem să numerotăm paginile clasorului de la sfârșit pentru eficiență. Dacă am numerota clasorul de la început paginile mai mici se vor regăsi în mai multe intervale, așa că noi numerotăm de la sfârșit ca să restrângem căutările până ajungem la pagina 1.

Cât timp nu am terminat de numerotat paginile, căutăm în vector toate valorile pentru care marginea superioară este mai mare sau egal cu nr. de pagini și ~~le~~ ~~de~~ adăugăm în coada costul. Când terminăm de parcurs vectorul adăugăm la o sumă elementul din vârful cozii (adică costul minim pt. că coada are elem. ord. crescător). Apoi scădem k din n pentru că mai avem de numerotat n-k pagini

Heapuri

Folosim un container de tip set (multime) și o structură de date vector - care va reține elem. distincte. Dacă citim un nr. îl adăugăm în structura set și în structura vector, dacă tipul operației este 1. Vectorul este necesar pentru operația de ștergere atunci când trebuie să eliminăm elem. de pe o poziție dată. Dacă tipul operației este 3 atunci aflăm elem. minim din set adică primul element pentru că în structura de date de tip set elem. sunt distincte și ord. crescător.

Hashuri

Folosim un array de vectori (structura de date folosită) care mapează (asociază) cheile la valori cunoscute. Când avem un număr, determinăm cheia ca fiind restul acelui număr la un număr prim mare pentru evitarea coliziunilor (asocierea mai multor valori aceleiași chei). Dacă tipul operației este 1, se caută valoarea citită în lista de valori care sunt asociate cheii valorii citite care are lungimea v[cheie].size() și dacă nu se găsește o adăugăm. Dacă tipul este 2, căutăm valoarea citită în lista de valori care sunt asociate cheii valorii citite și dacă o găsim ștergem valoarea din lista de valori asociate cheii respective de la poziția v[cheie].begin+j

Dacă tipul este 3 atunci căutăm valoarea citită în lista de valori asociate cheii valorii citite v[cheie].

Parcuri

Folosim o structură de date unordered_map care foloseste o combinație cheie-valoare. În cazul nostru cheia este timpul iar valoarea este suma de bani. Pentru fiecare pereche timp_bani retinem în container pentru timpul citit suma de bani citită și doar la un pas următor se citește pentru același moment de timp o altă sumă de bani se adaugă acea sumă de bani la cheia respectivă pentru a se cumula o sumă de bani pentru același moment de timp.

X.first → timpul

X.second → banii