Deque

Ruxandra-Laura Nanu

December 2021

1 Introducere

Deja am vazut cum structuri de date cum ar fi cozile sau stivele pot fi folositoare in rezolvarea diferitelor tipuri de probleme. Astazi voi introduce un nou tip de structura de date, numit deque.
Deque-ul permite urmatoarele operatii:

- Adaugarea unui element la **final** .
- Adaugarea unui element la **inceput** .
- Stergerea **primului** element .
- Stergerea **ultimului** element .

Si urmatoarele interogari:

- Aflarea valorii **primului** element .
- Aflarea valorii **ultimului** element .
- Verificare daca deque-ul contine un numar **nenul** de elemente .

In practica, deque-ul se poate simula manual cu ajutorul unui vector sau se poate folosi structura predefinita in STL.

2 Probleme usoare

2.1 Arhiva educationala - Deque

Problema este cel mai clasic exemplu de aplicatie al deque-ului.

In acest caz, pentru fiecare element ne intereseaza atat valoarea sa cat si indicele, pentru a determina daca am depasit cumva lungimea secventei cautate.

Mai intai, vom parcurge toate elementele sirului si vom folosi abordarea de la stiva pentru ca un element "mic" sa scoata alte elemente "mai mari" decat el care vin inainte.

Stiind ca putem afla valoarea doar primului element din deque, intervine urmatoarea problema: Ce se intampla daca vrem sa aflam minimul din secventa [i - k + 1 .. i], dar pe prima pozitie a deque-ului se afla un element care se afla pe o pozitie mai mica decat i - k + 1 (adica nu este in secventa care ne intereseaza).

Aici intervine operatia de stergere a primului element din deque, fiindca un element care se afla pe o pozitie din intervalul [1 ... i - k] nu va mai fi luat in calcul de niciuna dintre secventele de lungime k care se termina pe o pozitie mai mare sau egala cu i.

Deci, dupa ce ne-am asigurat ca primul element din deque apartine secventei pe care vrem sa o analizam, putem afla valoarea elementului si sa o adaugam la suma ceruta a minimelor.

Mai jos sunt doua surse la problema (una simuland deque-ul si alta folosind deque-ul din STL), alaturi de mai multe comentarii explicative:

Exemplu simulare deque
Exemplu deque STL

2.2 .campion 2005 - Vila2

Problema este asemanatoare cu cea anterioara, insa de data aceasta vrem, pentru fiecare secventa de lungime K, diferenta maxima in modul a doua elemente.

Observatia care conduce la solutia problemei este ca pentru o secventa, diferenta maxima in modul dintre oricare doua elemente este egala cu diferenta dintre maximul si minimul din secventa.

Din problema anterioara, stim cum sa aflam minimul din fiecare secventa de lungime K. Daca intr-un mod asemanator reusim sa obtinem, folosind alt deque, maximul din fiecare secventa de lungime K, putem cupla minimele cu maximele potrivite pentru a obtine diferenta maxima in valoare absoluta din fiecare secventa.

Exemplu de implementare aici

2.3 Stelele Informaticii 2003, clasele 11-12 - Sum2

Pentru a rezolva problema avem nevoie de o precalculare. Faptul ca se cer sumele pe secvente dintr-un sir ne duce cu gandul la calcularea sirului de sume partiale sp[i] = sp[i - 1] + v[i], unde v este sirul din input.

Cum transforma precalcularea felul in care ar trebui sa interpretam problema? Daca vrem sa verificam sumele tuturor secventelor valide care se incheie pe pozitia i, trebuie sa evaluam sp[i] - sp[j], unde i - u $\leq$ j $\leq$ i - l. Ca sa luam secventa de suma maxima de lungime valida si care se termina pe pozitia i, ar trebui sa maximizam suma sp[i] - sp[j], dar sp[i] este constanta (i este fixat), deci trebuie sa cautam j in intervalul convenabil astfel incat sp[j] este minim.

Deci, pentru un i, acum nu mai trebuie sa gasim minimul/maximul in intervalul [i - k + 1 .. i] (pentru un k dat) ci cautam minimul in intervalul [i - u .. i - l]. Totusi, diferenta asta este una minora, pentru ca putem sa folosim deque-ul la fel, schimband doar indicele elementului pe care incercam sa il adaugam la final si conditia pentru eliminarea primului element din deque.

Exemplu de implementare aici

3 Probleme medii

3.1 preONI 2007 Runda Finala - Branza

Problema se rezolva cu programare dinamica.

O observatie cheie este ca kilogramele de branza sunt independente intre ele; putem produce oricate in orice zi si putem stoca oricate in depozit, pretul de S unitati monetare fiind platit separat pe fiecare kilogram.

Asta inseamna ca putem calcula dp[i] = pretul minim cu care putem sa obtinem un kilogram de branza pe care sa il vindem in ziua i = min (C[j] + (i - j) * S) unde i - j $\leq$ T, din cauza restrictiei legata de numarul de zile timp de care putem stoca un kilogram.

Vrem sa minimizam dp[i] la fiecare pas, deci C[j] + (i - j) * S , i - j $\leq$ T. Expresia poate fi rescrisa ca : i * S + C[j] - j * S. De vreme ce i este fixat, trebuie doar sa cautam j (T - i $\leq$ j $\leq$ i) astfel incat C[j] - j * S sa fie minim. Putem gasi j-ul folosind un deque.

Solutia finala o sa fie dp[1] * P1 + dp[2] * P2 + .. + dp[n] * Pn, pentru ca vindem Pi kilograme in ziua i, iar productia unui kilogram vandut in ziua i poate costa minim dp[i].

Exemplu de implementare aici

3.2 preONI 2006 Runda 2 - Struti

Vom trata fiecare query independent.

Aflarea minimului pentru o submatrice cu coltul din dreapta jos in (i,j) si dimensiuni dx , dy se realizeaza similar cu aflarea maximului, deci o voi prezenta doar pe accea a maximului in material.

Presupunem ca matricea initiala este notata cu a

Mai intai, pentru un query vom calcula matricea maxi[i][j] = max(a[i][j] , a[i][j - 1] ... , a[i][j - dy + 1]) folosind un deque si luand liniile din a pe rand. Observam ca daca acum luam max(maxi[i][j] , maxi[i - 1][j] ... , maxi[i - dx + 1][j]), valoarea obtinuta va fi chiar maximul din submatricea de dimensiuni dx, dy cu coltul din dreapta jos in (i , j). La fel ca si pentru calcularea matricei 'maxi', pentru a obtine valoarea cautata putem sa mai folosim un deque, dar de data aceasta fixand cate o coloana si iterand prin linii.

Odata ce stim cum sa obtinem maximul dintr-o submatrice, putem proceda similar pentru a afla si minimul.

O observatie finala este ca trebuie sa 'rezolvam' un query atat pentru dimensiunile (dx , dy) cat si pentru (dy , dx), tinand cont de cazul particular cand dx = dy.

Exemplu de implementare aici

3.3 Lot Ploiesti 2006 - Drept2

Problema seamana cu cea anterioara, dar de data asta nu mai putem tine in memorie toata matricea din cauza dimensiunilor sale.

Pentru a determina cate drepunghiuri de dimensiune X , Y au latura de jos pe linia i, trebuie sa aflam y1 = capatul din stanga maxim al unei secvente de 1 care se afla pe una dintre ultimele X linii si, similar, y2 = capatul din dreapta minim al unei secvente de 1 care se afla pe una dintre ultimele X linii.

Odata ce pentru o linie i am calculat y1 si y2, solutia este y2 - y1 + 1 - Y + 1.

Inlocuind, deci (X,Y) cu (A,B) si (B,A) (in cazul cand A != B), putem obtine solutia. Avand in vedere solutia de la problema anterioara, devine evident cum doua deque-uri ne pot ajuta sa calculam valorile y1 si y2 la fiecare pas.

Exemplu de implementare aici

3.4 .campion 2007-2008, Runda 1 - Copaci2

O prima observatie este ca, presupunand ca daca o solutie X se poate obtine cu un buget K, orice solutie mai mare decat X va putea fi obtinuta cu un buget mai mic decat K. Cu alte cuvinte, cu cat cheltuim mai mult din buget cu atat diferenta maxima dintre doi copaci adiacenti scade.

Observatia asta inseamna ca putem cauta binar rezultatul si putem vedea daca cu bugetul nostru de K putem creste/scadea inaltimile copacilor astfel incat intre niciunii doi consecutivi diferenta absoluta a inaltimilor sa nu fie mai mare decat valoarea fixata.

Odata ce am hotarat sa vedem daca putem obtine o valoare, fie ea 'val' cu bugetul nostru de K, putem folosi dinamica dp[i][j] = costul minim asa incat copacul i sa ajunga cu inaltimea j.

Astfel, dp[i][j] = min(dp[i - 1][x] + abs(h[i] - j) * C) unde h[i] sunt inaltimile initiale ale copacilor iar abs(j - x) $\leq$ val. Adica luam in calcul doar potentiale inaltimi ale copacului de pe pozitia i - 1 asa incat sa se pastreze conditia impusa legata de diferenta absoluta dintre doi copaci consecutivi.

In final, trebuie sa vedem daca exista vreun dp[n][j] $\leq$ K, caz in care diferenta val poate fi obtinuta cu bugetul nostru. Altfel, trebuie sa cautam un 'val' mai mare.

Totusi, limita de memorie este mica, deci din matricea dp ar trebui tinute doar doua linii, de vreme ce pentru calcularea costurilor pentru copacul i ne folosim doar de valorile pentru copacul i-1.

Exemplu de implementare aici

4 Problema grea

4.1 Stelele Informaticii 2006, clasele 9-10 - Bcrc

Problema se rezolva folosind programare dinamica.

O stare consta din momentul curent de timp si camera in care ne aflam. Notam dp[i][j] = numarul de bomboane maxim pe care il putem obtine daca suntem in camera j la momentul i. Valorile se pot schimba doar atunci cand apare o noua cutie de bomboane, deci vom recalcula valorile din dp doar in M momente: de fiecare data cand apare o cutie noua.

Reinterpretam, deci semnificatia dp[i][j] = numarul de bomboane maxim pe care il putem obtine daca suntem in camera j in momentul in care a aparut

cutia i.

Avand in vedere ca stim care sunt 'optimele' pentru pozitiile in momentul in care apare cutia i-1 si ca in cele T = Ti - T(i-1) secunde intre aparitiile cutiilor i-1 si i putem doar sa ne deplasam in camere adiacente sau sa stam pe loc obtinem recurenta:

dp[i][j] = max(dp[i - 1][j - T] , dp[i - 1][j - T + 1] , ... , dp[i - 1][j + T])

Dupa calcularea acestor valori trebuie sa completam cu dp[i][Ci] += Bi

Pentru a calcula valorile din dp putem folosi un deque, si ca sa evitam cazurile particulare induse de pozitionarea camerelor in cerc in loc de liniar, putem sa facem o dublare fictiva a sirului. Adica dp[i][j] = dp[i][j + n], ca sa putem scadea sau adauga T fara sa depasim dimensiunile matricii dp.
Un caz particular important de notat este cand T > n/2, adica cand in intervalul de timp 'liber' putem ajunge in orice camera dorim.

Solutia se afla in max(dp[M][j]) pentru $1 \leq j \leq N$. Adica numarul maxim de bomboane obtinute dupa ce au aparut toate cutiile care ar fi putut aparea.

Exemplu de implementare aici

5 Mai multe probleme care se rezolva folosind deque-ul

- Gard
- Cover