

Deque

Deque

- Deque sau double ended queue este o structura de date similara cozii si stivei. Este oarecum o generalizare a acestora.

Deque

- Deque sau double ended queue este o structura de date similara cozii si stivei. Este oarecum o generalizare a acestora.
- Structura ne permite adaugarea si eliminarea elementelor atat de la capat cat si de la coada.

Deque

- Deque sau double ended queue este o structura de date similara cozii si stivei. Este oarecum o generalizare a acestora.
- Structura ne permite adaugarea si eliminarea elementelor atat de la capat cat si de la coada.
- In C++ avem biblioteca “deque” care implementeaza pentru noi structura de deque (deci nu trebuie sa ne batem capul cu asta).

Deque - Aplicatie

- O aplicatie ce se foloseste de un deque (impreuna cu explicatiile) poate fi gasita la acest [link](#).

Deque - Aplicatie

- O aplicatie ce se foloseste de un deque (impreuna cu explicatiile) poate fi gasita la acest [link](#).
- Rezumat: Se cere sa se afle minimul pentru fiecare subsecventa de lungime K si apoi sa se afle suma acestor minime.

Deque - Aplicatie

- O aplicatie ce se foloseste de un deque (impreuna cu explicatiile) poate fi gasita la acest [link](#).
- Rezumat: Se cere sa se afle minimul pentru fiecare subsecventa de lungime K si apoi sa se afle suma acestor minime.
- Solutie bruta: Iteram prin toate subsecventele de lungime K , obtinem minimul si il adaugam in suma totala.

Deque - Aplicatie

- O aplicatie ce se foloseste de un deque (impreuna cu explicatiile) poate fi gasita la acest [link](#).
- Rezumat: Se cere sa se afle minimul pentru fiecare subsecventa de lungime K si apoi sa se afle suma acestor minime.
- Solutie bruta: Iteram prin toate subsecventele de lungime K, obtinem minimul si il adaugam in suma totala.
- Complexitate: ?

Deque - Aplicatie

- O aplicatie ce se foloseste de un deque (impreuna cu explicatiile) poate fi gasita la acest [link](#).
- Rezumat: Se cere sa se afle minimul pentru fiecare subsecventa de lungime K si apoi sa se afle suma acestor minime.
- Solutie bruta: Iteram prin toate subsecventele de lungime K, obtinem minimul si il adaugam in suma totala.
- Complexitate: $O((N - K) * K)$

Ideea de optimizare

- Sa presupunem ca am calculat minimul si pozitia acestuia pe subsecventa de la i la $i + K - 1$.

Ideea de optimizare

- Sa presupunem ca am calculat minimul si pozitia acestuia pe subsecventa de la i la $i + K - 1$.
- Avem cateva cazuri:

Ideea de optimizare

- Sa presupunem ca am calculat minimul si pozitia acestuia pe subsecventa de la i la $i + K - 1$.
- Avem cateva cazuri:
 - Minimul se afla pe pozitia i

Ideea de optimizare

- Sa presupunem ca am calculat minimul si pozitia acestuia pe subsecventa de la i la $i + K - 1$.
- Avem cateva cazuri:
 - Minimul se afla pe pozitia i
 - Minimul se afla in subsecventa de la $i + 1$ la $i + K - 1$

Ideea de optimizare

- Sa presupunem ca am calculat minimul si pozitia acestuia pe subsecventa de la i la $i + K - 1$.
- Avem cateva cazuri:
 - Minimul se afla pe pozitia i
 - Minimul se afla in subsecventa de la $i + 1$ la $i + K - 1$
- Sa mutam acum capetele secventei, de la $i + 1$ la $i + K$.

Ideea de optimizare

- Sa presupunem ca am calculat minimul si pozitia acestuia pe subsecventa de la i la $i + K - 1$.
- Avem cateva cazuri:
 - Minimul se afla pe pozitia i
 - Minimul se afla in subsecventa de la $i + 1$ la $i + K - 1$
- Sa mutam acum capetele secventei, de la $i + 1$ la $i + K$. Minimul este fie minimul de pe subsecventa de la $i + 1$ la $i + K - 1$, fie elementul $i + K$.

Ideea de optimizare

- Sa presupunem ca am calculat minimul si pozitia acestuia pe subsecventa de la i la $i + K - 1$.
- Avem cateva cazuri:
 - Minimul se afla pe pozitia i
 - Minimul se afla in subsecventa de la $i + 1$ la $i + K - 1$
- Sa mutam acum capetele secventei, de la $i + 1$ la $i + K$. Minimul este fie minimul de pe subsecventa de la $i + 1$ la $i + K - 1$, fie elementul $i + K$.
- Observam ca minimul din subsecventa precedenta ar putea sa fie refolosit (in anumite cazuri).

Solutia isteata

- Vom retine “candidatii” la minim intr-un deque.

Solutia isteata

- Vom retine “candidatii” la minim intr-un deque.
- Cand vom muta secventa la dreapta:

Solutia isteata

- Vom retine “candidatii” la minim intr-un deque.
- Cand vom muta secventa la dreapta:
 - vom elimina elementul de pe pozitia $i - K$ (daca este in deque)

Solutia isteata

- Vom retine “candidatii” la minim intr-un deque.
- Cand vom muta secventa la dreapta:
 - vom elimina elementul de pe pozitia $i - K$ (daca este in deque)
 - vom elimina toate elementele ce nu mai pot fi minime (elementul i este mai mic)

Solutia isteata

- Vom retine “candidatii” la minim intr-un deque.
- Cand vom muta secventa la dreapta:
 - vom elimina elementul de pe pozitia $i - K$ (daca este in deque)
 - vom elimina toate elementele ce nu mai pot fi minime (elementul i este mai mic)
 - vom adauga elementul i in deque

Solutia isteata

- Vom retine “candidatii” la minim intr-un deque.
- Cand vom muta secventa la dreapta:
 - vom elimina elementul de pe pozitia $i - K$ (daca este in deque)
 - vom elimina toate elementele ce nu mai pot fi minime (elementul i este mai mic)
 - vom adauga elementul i in deque
 - minimul pe secventa de la $i - K + 1$ la i este minimul din deque.

Deque - Utilizare

- Declarare: `std::deque<tip_data_stocat> nume_deque;`

Deque - Utilizare

- Declarare: `std::deque<tip_data_stocat> nume_deque;`
- Adaugarea unui element in cap, respectiv coada:
`nume_deque.push_front(x); nume_deque.push_back(x);`

Deque - Utilizare

- Declarare: `std::deque<tip_data_stocat> nume_deque;`
- Adaugarea unui element in cap, respectiv coada:
`nume_deque.push_front(x); nume_deque.push_back(x);`
- Eliminarea unui element din cap, respectiv coada:
`nume_deque.pop_front(); nume_deque.pop_back();`

Deque - Utilizare

- Declarare: `std::deque<tip_data_stocat> nume_deque;`
- Adaugarea unui element in cap, respectiv coada:
`nume_deque.push_front(x); nume_deque.push_back(x);`
- Eliminarea unui element din cap, respectiv coada:
`nume_deque.pop_front(); nume_deque.pop_back();`
- Extragerea elementului din cap, respectiv coada (nu se elimina): `nume_deque.front(); nume_deque.back();`

Deque - Utilizare

- Declarare: `std::deque<tip_data_stocat> nume_deque;`
- Adaugarea unui element in cap, respectiv coada:
`nume_deque.push_front(x); nume_deque.push_back(x);`
- Eliminarea unui element din cap, respectiv coada:
`nume_deque.pop_front(); nume_deque.pop_back();`
- Extragerea elementului din cap, respectiv coada (nu se elimina): `nume_deque.front(); nume_deque.back();`
- Verificare daca este goala si lungimea se pot obtine similar ca pentru o coada/stiva normala (functiile `empty` si `size`).

Tema

- Alte probleme rezolvate pe ideea precedenta:
 - Vila2
 - Secventa
 - Branza
 - Cuie
 - Struti
- Problema cu deque ce nu se foloseste de ideea aceasta: KNumere
- Alte aplicatii si materiale de citit (desi contin si subiecte mai dificile) deque-si-aplicatii