THUẬT TOÁN ỨNG DỤNG

CẤU TRÚC DEQUE

Phạm Quang Dũng
Bộ môn KHMT
dungpq@soict.hust.edu.vn

DEQUE

- Cấu trúc dữ liệu tuyến tính có tính chất của cả ngăn xếp và hàng đợi:
  - Thêm 1 phần tử vào cuối deque
  - Lấy 1 phần tử ở đầu deque ra
  - Lấy 1 phần tử ở cuối deque ra
- Trong C++
  - Khai báo: deque<int>
  - Phương thức: push_back(), push_front(), pop_front(), pop_back(), back(), front(), empty()

Quy hoạch động sử dụng deque

- Cho dãy a_1, a_2, …, a_n và 2 số nguyên dương $L_1 < L_2$. Hãy tìm dãy con $1 \le j_1 < j_2 < \ldots < j_k \le n$ sao cho $L_1 \le j_{q+1} - j_q \le i_2$ và a_{j_1}, a_{j_2}, …, a_{j_k} có tổng cực đại

Quy hoạch động sử dụng deque

- Định nghĩa bài toán con
  - $S(i)$: tổng cực đại của dãy con của dãy $a_1, \ldots, a_i$ thỏa mãn đề bài mà phần tử cuối cùng là a_i
- Công thức quy hoạch động
  - $S(i) = \max(a_i + S(j) \mid L_1 \le i - j \le L_2\}$

Quy hoạch động sử dụng deque

- Định nghĩa bài toán con
  - $S(i)$: tổng cực đại của dãy con của dãy $a_1, \ldots, a_i$ thỏa mãn đề bài mà phần tử cuối cùng là a_i
- Công thức quy hoạch động
  - $S(i) = \max(a_i + S(j) \mid L_1 \le i - j \le L_2\}$
- Khởi tạo deque, lưu trữ các chỉ số j sao cho $S(j)$ không tăng và là ứng cử viên để tính toán các bài toán con $S(i)$
- Mỗi khi xét đến chỉ số i ($i = 1,\ldots, n$) thì
  - Đưa hết các chỉ số j ở đầu deque tại đó $j < i - L_2$ ra ngoài (vì nó ko là ứng cử viên để xác định S(i), S(i+1),…)
  - Đưa hết các chỉ số j ở cuối deque tại đó $S(j) < S(i\text{-}L_1)$ (do những chỉ số j như vậy không có ý nghĩa nữa trong việc xác định $S(i)$, $S(i+1)$,…

Quy hoạch động sử dụng deque

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1e6+1;
int a[N], S[N];
int n,L1,L2,ans;
int main(){
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    cin >> n >> L1 >> L2;
    for(int i = 1; i <= n; i++)
       cin >> a[i];
    deque<int> q;
    ans = 0;
```

```
 for(int i = 1; i <= n; i++){
  while(!q.empty() && (q.front() <
                                  i - L2))
      q.pop_front();
  if(i - L1 >= 1){
     while(!q.empty() && S[q.back()] <
             S[i- L1])
       q.pop_back();
     q.push_back(i-L1);
   }
   S[i] = a[i] + (q.empty() ? 0 :
                           S[q.front()]);
   ans = max(ans,S[i]);
 }
 cout << ans;
}
```