HUST

ĐẠI HỌC BÁCH KHOA HÀ NỘI

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

ONE LOVE. ONE FUTURE.

ĐẠI HỌC
BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY
OF SCIENCE AND TECHNOLOGY

Applied Algorithm Lab

Gold mining

ONE LOVE. ONE FUTURE.

Gold mining

- Given n gold warehouse in a straight line
  - Warehouse i stores an amount of a_i and located at point i in the line
- **Objective:** Find a subset of warehouse with largest sum of amount
- **Constraint:** distance between 2 warehouses must be in $[\mathrm{L1}, \mathrm{L2}]$
- **Output:** The amount found
- **Example:**

Input	**Output**
6 2 3	19
3 5 9 6 7 4	explain: 3+9+7=19

- **Idea to solve #1:** Backtracking and Branch and bound
  - List all ways to choose a subset of gold warehouse
    - For each case, check if 2 consecutive warehouse has distance in [L1, L2]
    - if all pair satisfy distance constraint, update the amount if needed
  - Complexity: O($2^n * n$).
  - Some BnB technique can be applied:
    - While considering the i warehouse, the next ware house must be in [i + L1, i + L2].
    - Apply BnB for objective function
    - still can not pass full testcases…

- **Idea to solve #2:** Dynamic programming: Consider choosing warehoue i
  - Let F[i] be maximal amount available if we choose some warehouses from 1 to i-1 **and choose warehouse i.**
  - Base case: F[i] = a[i].
  - Formula:

$$\mathrm{F}[\mathrm{i}] = \ max_{j\epsilon[i\ -L2,\ i\ -L1]}(a[i] + F[j]), \forall i\epsilon[L1, n).$$

  - return:

$$max_i F[i], \forall i \in [1, n].$$

  - Complexity: O(N^2).

- **Idea to solve #3:** Dynamic programming with priority queue
  - Priority queue: a queue with order. The element are sorted in order of priority
  - Improve to idea #2:
    - An element in priority queue: (j, F[j]) with F[j] be the priority.
    - element with big F[j] will be in the front
    - Considering ware house i: add (i – L1, F[i – L1]) to the queue.
    - Remove the top element of queue if i – i.top > L2.
    - F[i] = a[i] + F.top.
  - Complexity: O(n*log(n)).

Gold mining

- Idea to solve #2: Dynamic programming with dequeue
  - Dequeue: a combination of stack and queue -> element can be add or remove in both the back and front of dequeue
  - Operation: push_back(), push_front(), pop_back(), pop_front()
  - Improvement: Element in queue: (x, F[x]). Traverse F[i] in order.
    - Remove all element j that $F[j] \leq F[i - L1]$, then add F[i – L1] to the queue.
    - Delete the top element of the queue until top >= i – L2.
    - F[i] = F[top] + a[i].
  - Complexity: O(n).

Gold mining - Implementation

```
#include <bits/stdc++.h>
using namespace std;
#define N 1000001
int n, L1, L2, ans;
int a[N];
int S[N];
void solve(){
    // luu tru chi so cac ung cu vien j tham gia vao xac dinh bai toan con S[j]
    deque<int> d;
    ans = 0;
    for (int i = 1; i <= n; i++){
        while (!d.empty() && (d.front() < i - L2))
            d.pop_front();
        int j = i - L1;
        if (j >= 1){
            while (!d.empty() && (S[d.back()] < S[j]))
                d.pop_back();
            d.push_back(j);
        }
        S[i] = a[i] + (d.empty() ? 0 : S[d.front()]);
        ans = max(ans, S[i]);
    }
}
```

Gold mining - Implementation

```
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];
    solve();
    cout << ans;
    return 0;
}
```

HUST

hust.edu.vn fb.com/dhbkhn

THANK YOU !