

Gold Mining

- The Kingdom ALPHA has n warehouses of golds located on a straight line and are numbered $1, 2, \dots, n$. The warehouse i has amount of a_i (a_i is non-negative integer) and is located at coordinate i ($i = 1, \dots, n$). The King of ALPHA opens a competition for hunters who are responsible to find a subset of gold warehouses having largest total amount of golds with respect to the condition that the distance between two successive selected warehouses must be greater than or equal to L_1 and less than or equal to L_2 .
- **Input**
 - Line 1 contains n , L_1 , and L_2 ($1 \leq n \leq 100000, 1 \leq L_1 \leq L_2 \leq n$)
 - Line 2 contains n integers a_1, a_2, \dots, a_n
- **Output**
 - Contains only one single integer denoting the total amount of golds of selected warehouses.
- **Example:**
- **Input**

6 2 3

3 5 9 6 7 4

- **Output**

19

Đề bài:

Có n nhà kho nằm trên một đường thẳng.

Nhà kho i nằm ở vị trí i và có số lượng vàng là a_i

Yêu cầu:

Chọn các nhà kho sao cho:

Tổng lượng vàng là lớn nhất.

2 nhà kho liên tiếp có khoảng cách nằm trong khoảng $[L_1, L_2]$.

Gold Mining – Backtracking Algorithm

- Duyệt hết tất cả các trường hợp chọn các nhà kho khác nhau:
- Với mỗi trường hợp, kiểm tra xem 2 nhà kho liên tiếp có khoảng cách nằm trong khoảng $[L1, L2]$ hay không, nếu tất cả các nhà kho đều thỏa mãn thì cập nhật tổng lượng vàng.
- Độ phức tạp: $O(2^n * n)$.
- Có thể thực hiện một số biện pháp nhánh cận như:
 - Khi đang xét đến nhà kho thứ i , cân nhắc chỉ xét các nhà kho trong đoạn $[i + L1, i + L2]$.

Gold Mining – Dynamic Programming Algorithm $O(N^2)$

- Gọi $F[i]$ là tổng lượng vàng lớn nhất nếu chọn các nhà kho từ 1 đến $i - 1$ và nhà kho thứ i được chọn.
- Khởi tạo: $F[i] = a[i]$.
- Công thức:

$$F[i] = \max_{j \in [i-L_2, i-L_1]} (a[i] + F[j]), \forall i \in [L_1, n].$$

- Kết quả:

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

Độ phức tạp: $O(N^2)$.

Gold Mining – Dynamic Programming Algorithm ($O(n)$)

- Hàng đợi 2 đầu (deque) là cấu trúc dữ liệu kết hợp giữa hàng đợi và ngăn xếp -> phần tử đều có thể được thêm vào và lấy ra ở đầu và ở cuối deque.
- Thao tác: push_back(), push_front(), pop_back(), pop_front()
- Cải tiến: Các phần tử trong hàng đợi là chỉ số j tham gia vào ứng viên xác định $F[i]$.
 - Duyệt $F[i]$ theo thứ tự $i = 2, 3, \dots, n$.
 - Xóa mọi phần tử j mà $F[j] \leq F[i - L1]$ trong hàng đợi, thêm chỉ số $i - L1$ vào hàng đợi.
 - Xóa phần tử đầu tiên top của hàng đợi cho đến khi $\text{top} \geq i - L2$.
 - $F[i] = F[\text{top}] + a[i]$.

Gold Mining – Dynamic Programming Algorithm O(N^2)

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1e6+1;
int a[N], S[N];
int n, L1, L2, ans;
void input(){
    ios_base::sync_with_stdio(0); cin.tie(0);
    cin >> n >> L1 >> L2;
    for(int i = 1; i <= n; i++) cin >> a[i];
}
void solveN2(){
    S[1] = a[1];ans = S[1];
    for(int i = 2; i <= n; i++){
        S[i] = a[i];
        for(int d = L1; d <= L2; d++){
            int j = i-d;
            if(j >= 1 && S[i] < S[j] + a[i]) S[i] = S[j] + a[i];
        }
        ans = max(ans,S[i]);
    }
    cout << ans;
}
int main(){
    input();
    solveN2();
    return 0;
}
```

Implementation – use deque (or vector)

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1e6+1;
int a[N], S[N];
int n, L1, L2, ans;

void solve(){
    deque<int> d;// luu tru chi so cac ung cu vien j tham gia vao viec xac dinh cac bai toan
                  // con S(i)
    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]);
    }
    cout << ans;
}
```

Implementation – use deque (or vector)

```
void input(){
    ios_base::sync_with_stdio(0); cin.tie(0);
    cin >> n >> L1 >> L2;
    for(int i = 1; i <= n; i++) cin >> a[i];
}
int main(){
    input();
    solve();
    return 0;
}
```