Templates (Mai Đức An)

オーナー	Tiểu Phương
タグ	

Binary Search Tree

```
typedef struct Node {
    int val;
    Node* left;
    Node* right;
} Node;

Node* newNode(int v) {
    Node* node = new Node;
    node->val = v;
    node->left = node->right = NULL;
    return node;
}

Node* insert(Node* root, int v) {
    if (root == NULL) return newNode(v);
    if (v < root->val) root->left = insert(root->left, v);
    else root->right = insert(root->right, v);
}

Node* findMax(Node* root) {
    if (root == NULL) return NULL;
    if (root->right == NULL) return root;
    else return findMax(root->right);
}

Node* remove(Node* root, int v) {
    if (root == NULL) return root;
```

```
    if (v < root->val) root->left = remove(root->left, v);
    else if (v > root->val) root->right = remove(root->right,
    else {
        if (root->left == NULL) {
            Node* temp = root->right;
            free(root);
            return temp;
        }
        else if (root->right == NULL) {
            Node* temp = root->left;
            free(root);
            return temp;
        }
        Node* temp = findMax(root->left);
        root->val = temp->val;
        root->left = remove(root->left, temp->val);
    }
}
```

Binary Search - Lower Bound

```
int lowerBoundBS(int arr[], int size, int target) {
    int left = 0;
    int right = size;
    while (left < right) {
        int mid = left + (right - left) / 2;
        if (arr[mid] < target) {
            left = mid + 1;
        }
        else {
            right = mid;
        }
    }
    return left;
}
```

Binary Search - Upper Bound

```
int upperBoundBS(int arr[], int size, int target) {
    int left = 0;
    int right = size;
    while (left < right) {
        int mid = left + (right - left) / 2;
        if (arr[mid] <= target) {
            left = mid + 1;
        }
        else {
            right = mid;
        }
    }
    return left;
}
```

Longest Increasing Subsequence

```
cin >> n;
for (int i=1; i<=n; i++) cin >> a[i];
for (int i=1; i<=n; i++) {
    dp[i] = 1;
    for (int j=0; j<i; j++) {
        if (a[j] < a[i]) dp[i] = max(dp[i], dp[j] + 1);
    }
}
for (int i=1; i<=n; i++) ans = max(ans, dp[i]);
cout << ans;
```

Longest Common Subsequence

```
cin >> n >> m;
for (int i=1; i<=n; i++) cin >> x[i];
for (int i=1; i<=m; i++) cin >> y[i];
for (int i=1; i<=n; i++) {
    for (int j=1; j<=m; j++) {
        if (x[i] == y[j])
            dp[i][j] = dp[i-1][j-1] + 1;
        else dp[i][j] = max(dp[i-1][j], dp[i][j-1]);
```

```
    }
}
cout << dp[n][m];
```

Knapsack

```
cin >> s >> n; // n tui
for (int i=1; i<=n; i++) cin >> w[i] >> v[i];
for (int i =1; i<=n; i++) {
    for (int j=0; j<=s; j++) {
        dp[i][j] = dp[i-1][j];
        if (j >= w[i]) {
            dp[i][j] = max(dp[i][j], dp[i-1][j-w[i]] + v[i]);
        }
    }
}
cout << dp[n][s];
```

Tổng dãy con lớn nhất 3 ptu ko liên tiếp

```
cin >> n;
for (int i = 1; i <= n; i++) {
    cin >> a[i];
}
dp[1] = a[1];
dp[2] = a[1] + a[2];
for (int i = 3; i <= n; ++i) {
    dp[i] = max({dp[i-1], dp[i-2] + a[i], dp[i-3] + a[i-1] +
}
cout<< dp[n];
```

Cbus

```
#include<bits/stdc++.h>
using namespace std;

#define ll long long
#define MAX 100
```

```
int N; // number of requests (1,2,...,N). Request i has picku
int cap; // number of places of the bus
int A[2*MAX+1][2*MAX+1];

int x[MAX];
int appear[MAX]; // marking
int load;
int f;
int f_best;
int x_best[MAX];
int cmin;

void input() {
    scanf("%d%d", &N, &cap);
    cmin = 10000000;
    for (int i = 0; i <= 2*N; i++) {
        for (int j = 0; j <= 2*N; j++) {
            scanf("%d", &A[i][j]);
            if (i != j && cmin > A[i][j]) cmin = A[i][j];
        }
    }
}

int check(int v, int k) {
    if (appear[v] == 1) return 0;
    if (v > N ) {
        if (!appear[v-N] == 1) return 0;
    } else {
        if (load + 1 > cap) return 0;
    }
    return 1;
}

void solution() {
    if (f + A[x[2*N]][0] < f_best) {
        f_best = f + A[x[2*N]][0];
        for (int i = 0; i <= 2*N; i++) x_best[i] = x[i];
```

```
    }
}

void TRY(int k) {
    for (int v = 1; v <= 2*N; v++) {
        if (check(v,k)) {
            x[k] = v;
            f += A[x[k-1]][x[k]];
            if (v <= N) load += 1; else load += -1;
            appear[v] = 1;
            if (k == 2*N) solution();
            else {
                if (f + (2*N+1-k) * cmin < f_best)
                    TRY(k+1);
            }
            if (v <= N) load -= 1; else load -= -1;
            appear[v] = 0;
            f -= A[x[k-1]][x[k]];
        }
    }
}

void solve() {
    load = 0;
    f = 0;
    f_best = 1000000;
    for (int i = 1; i <= 2*N; i++) appear[i] = 0;
    x[0] = 0; // starting point
    TRY(1);
    cout << f_best;
}

int main()
{
    ios_base::sync_with_stdio(0);
    cin.tie(0);cout.tie(0);
    input();
    solve();
```

```
    return 0;
}
```

Độ cao cao nhất, độ sâu sâu nhất của dãy số

```
cin >> n;
for (int i=0; i<n; i++) cin >> a[i];
int l_foot = 0, r_foot=0;
int head = 0;
int height=0, deepth = 0;
for (int i=1; i<n; i++) {
    if (a[i]>a[i-1]) {
        head=i;
        if (r_foot == i-1) l_foot = i-1;
    }
    if (a[i]<a[i-1]) r_foot=i;
    if (a[i] == a[i-1]) l_foot=r_foot=head=i;
    if (l_foot<head && head<r_foot) height = max (height, min
}

for (int i=1; i<n; i++) {
    if (a[i]<a[i-1]) {
        head=i;
        if (r_foot == i-1) l_foot = i-1;
    }
    if (a[i]>a[i-1]) r_foot=i;
    if (a[i] == a[i-1]) l_foot=r_foot=head=i;
    if (l_foot<head && head<r_foot) deepth = max (deepth, min
}
cout<<height << " " << deepth;
```

Điền dãy ngoặc (??(??))

```
// '('= +1, ')'= -1, '()' = 0
// dp[i][j] xét phần tử i có tổng điểm j
int solve(string s) {
    if (s[0] == ')') return 0;
    if (s[0] == '(') dp[0][1] = 1;
```

```
    if (s[0] == '?') dp[0][1] = 1;
    for (int i=1; i<s.length(); i++) {
        for (int j=0; j<= i; j++) {
            if (s[i] == '(') {
                if(j>0) dp[i][j] += dp[i-1][j-1] % MOD;
            }
            if (s[i] == ')')
                dp[i][j] += dp[i-1][j+1] % MOD;
            if (s[i] == '?') {
                dp[i][j] += dp[i-1][j+1] % MOD;
                if (j>0) dp[i][j] += dp[i-1][j-1] % MOD;
            }
        }
    }
    return dp[s.length()-1][0];
}
```

Đếm số nghiệm nguyên $a_1X_1 + a_2X_2 + \ldots + a_nX_n = M$

```
int n, m;
int x[MAX], A[MAX], t[MAX];
int ans = 0;
int sum = 0;
void solution()
{
    ans++;
}
void Try(int k)
{
    for(int v = 1; v <= (m-sum - (t[n]-t[k])/A[k]); v++)
    {
        x[k] = v;
        sum += x[k] * A[k];
        if(k == n)
        {
            if(sum == m)
                solution();
        }
```

```
        else
            Try(k + 1);
        sum -= x[k] * A[k];
    }
}
int main()
{
    ios_base::sync_with_stdio(0);
    cin.tie(0);cout.tie(0);
    cin >> n >> m;
    for(int i = 1; i <= n; i++)
        cin >> A[i];
    t[0] = A[0];
    for (int i = 1; i < n; i++)
        t[i] = t[i - 1] + A[i];
    Try(1);
    cout << ans;
    return 0;
}
```

MaxEvenSubsequence

```
cin >> n;
for (int i = 1; i <= n; i++) cin >> a[i];
if (a[1] % 2) {
    S1[1] = a[1];
    B1[1] = true; B0[1] = false;
}
else {
    S0[1] = a[1];
    B1[1] = false; B0[1] = true;
}
for(int i = 2; i <= n; i++){
    if(a[i]%2 == 0){
        if(B0[i-1]){
            if(S0[i-1] >0) S0[i] = S0[i-1] + a[i];
            else S0[i] = a[i]; B0[i] = true;
        }
```

```
        else {
            S0[i] = a[i];
            B0[i] = true;
        }
        if(B1[i-1]){
            S1[i] = S1[i-1] + a[i];
            B1[i] = true;
        }
        else{
            B1[i] = false;
        }
    }else{
        if(B1[i-1]){
            S0[i] = S1[i-1] + a[i];
            B0[i] = true;
        }
        else{
            B0[i] = false;
        }
        if(B0[i-1]){
            if(S0[i-1] > 0) S1[i] = S0[i-1] + a[i];
            else S1[i] = a[i];  B1[i] = true;
        }
        else{
            S1[i] = a[i]; B1[i] = true;
        }
    }
}
ll ans = INF;
for (int i=1; i<=n; i++)
    if (B0[i])
        ans = max(ans, S0[i]);
cout << ans;
```