1. **Динамичко програмирање**

Дат је низ целих бројева који има N ($2 \leq N \leq 10^5$) чланова. Написати програм који ће исписати највећу суму подниза који има паран број елемената. Подниз је подскуп узастопних елемената низа.

Прва линија стандардног улаза садржи цео број N. Друга линија садржи N целих бројева (између 10^{-9} и 10^9) који представљају низ. На стандардном излазу одштампајте један цео број који представља максималну суму подниза парне дужине.
Временска сложеност O(n)

Улаз	**Излаз**	**Појашњење**
3 1 2 3	5	[2, 3]
5 1 -4 2 -16 8	-2	[-4, 2]
5 8 9 -8 9 10	20	[9, -8, 9, 10]

Решење:

```
#include <algorithm>

#include <iostream>

#include <vector>

using namespace std;

int main() {

   int n; cin >> n;

   vector<int> elements(n, 0);

   for (auto& itr : elements) {

      cin >> itr;

   }
```

```
    Long long mx = -2e9;
    vector<long long> dp(n + 2, 0);

    for (int i = n - 2; i >= 0; i -= 1) {
        dp[i] = elements[i] + elements[i + 1] + dp[i + 2];
        mx = max(mx, dp[i]);
        dp[i] = max(0LL, dp[i]);
    }

    cout << mx << '\n';
    return 0;
}
```

2. **Граф**

Вероватно да ово нисте знали, али у нашој галаксији има N настањених планета. Између неких планета постоје устаљени међупланетарни летови. Са сваке планете могуће је доћи до било које друге и ако постоји лет од планете A до планете B, постоји и повратни лет и он кошта C динара. Велики трошкови одржавања међупланетарних линија довели су до тога да је галактички одбор за транспорт одлучио да укине све летове који неће реметити повезаност свих планета, али тако да се трошкови одржавања свих преосталих летова сведу на минимум.

Улаз

У првој линији стандардног улаза налазе се два цела броја N и M. (број планета у галаксији $1 \leq N \leq 5000$ и број међупланетарних летова $N-1 \leq M \leq 100000$). У следећих M редова налазе се по три броја A, B и C одвојени размаком. A и B су бројеви планета које су повезане, а C је цена одржавања међупланетарне линије између планета A и B. ($1 \leq A,B,C \leq 5000$)

Излаз

У прву линију стандардног излаза испишите један цео број који представља цену одржавања преосталих летова након укидања оних неповољних.

Временска сложеност алгоритма: $O(M \log M)$

Меморијска сложеност: $O(M \log N)$.

Пример

Улаз

5 10

2 3 2

2 1 3

3 5 3

3 4 16

2 4 10

5 1 15

4 5 2

2 5 19

1 3 9

4 1 10

Излаз

10

Решење:

Применити МЦСТ алгоритам Kruscal, Prim,...

KRUSCAL

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

const int N = 5e3 + 10;
const int M = 1e5 + 10;

struct Edge {
    int u, v, w;

    bool operator < (const Edge &other) {
        return w < other.w;
    }
} edges[M];

int dsu[N];

void init(int n) {
    for (int i = 0; i < n; i++)
        dsu[i] = i;
}

int find_set(int x) {
    if (dsu[x] == x)
        return x;
    return dsu[x] = find_set(dsu[x]);
}

void unite_set(int x, int y) {
    dsu[find_set(x)] = dsu[find_set(y)];
}

long long Kruskal(int m) {
```

```
    long long mst = 0;
    sort(edges, edges + m);

    for (int i = 0; i < m; i++) {
        int u = edges[i].u, v = edges[i].v, w = edges[i].w;
        if (find_set(u) != find_set(v)) {
            unite_set(u, v);
            mst += 1ll * w;
        }
    }

    return mst;
}

int main() {
    int n, m;
    scanf("%d%d", &n, &m);
    for (int i = 0; i < m; i++)
        scanf("%d%d%d", &edges[i].u, &edges[i].v, &edges[i].w);

    init(n);

    printf("%lld\n", Kruskal(m));

    return 0;
}
```

PRIM

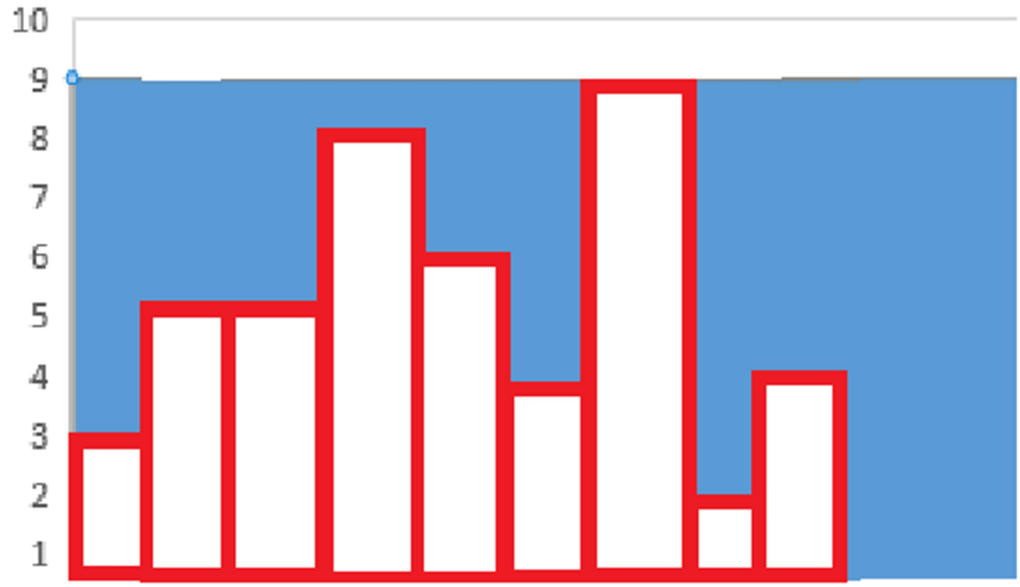
```
#include <bits/stdc++.h>


using namespace std;

struct graf{

    int n;
    vector<vector<pair<int,int>>> susedi;
    vector<int> roditelj;
    vector<int> rastojanja;
    vector<bool> posecen;
    priority_queue<pair<int, int>, vector<pair<int, int>>,
greater<pair<int,int>>> heap;

    graf(int n){

        susedi.resize(n);
        roditelj.resize(n, -1);
        rastojanja.resize(n, numeric_limits<int>::max());
        posecen.resize(n, false);
    }

    void dodaj(int a, int b, int c){
```

```
        susedi[a].push_back(make_pair(b, c));
        susedi[b].push_back(make_pair(a, c));
    }

    void prim(){

        rastojanja[0] = 0;
        heap.push(make_pair(0, 0));
        int ukupno = 0;

        while(!heap.empty()){

            int rastojanje = heap.top().first;
            int cvor = heap.top().second;
            heap.pop();

            if(posecen[cvor])
                continue;

            posecen[cvor] = true;
            ukupno += rastojanje;

            for(auto i : susedi[cvor]){

                int sused = i.first;
                int tezina = i.second;

                if(!posecen[sused] && tezina < rastojanja[sused]){

                    rastojanja[sused] = tezina;
                    roditelj[sused] = cvor;
                    heap.push(make_pair(tezina, sused));
                }
            }
        }

        cout << ukupno << endl;
    }
};

int main(){

    int n, m;
    cin >> n >> m;

    graf g = graf(n);

    for(int i=0; i<m; i++){

        int a, b, c;
        cin >> a >> b >> c;
        g.dodaj(a-1, b-1, c);
    }

    g.prim();
```

```
    return 0;
}
```

3. **Стек**

Низ бројева представља висине стубића у хистограму (сваки стубић је јединичне сирине). Одреди површину највећег правоугаоника у том хистограму.



Улаз: Први ред стандардног улаза садржи цео број N ($N \leq 1000000$). У следећем реду је дато N целих бројева $H_1, H_2, \ldots, H_N$, где H_I је висина I-тог стубића, $0 < H_I \leq 15000$.

Излаз: Програм треба да испише на стандардни излаз нађену максималну површину.

Временска сложеност алгоритма: $O(N)$

УЛАЗ
9
3 5 5 8 6 4 9 2 4
ИЗЛАЗ
24

Решење: са часа
http://poincare.matf.bg.ac.rs/~jelenagr/2017kiaaRsmer/StekNajveciPravougaonikHistogram.pdf