Contents

[Train 1 2](#_Toc45305109)

[B.01 2](#_Toc45305110)

[Train2 3](#_Toc45305111)

[A. Hist – Tìm diện tích hình chữ nhật max 3](#_Toc45305112)

[B. Postman 5](#_Toc45305113)

[C. Locate 7](#_Toc45305114)

[D. WATERJUG BFS 10](#_Toc45305115)

[E. REROAD 11](#_Toc45305116)

[F. SIGNAL 13](#_Toc45305117)

[G. Dễ 15](#_Toc45305118)

[H. PARENTHESES 15](#_Toc45305119)

[TRAIN 3 17](#_Toc45305120)

[A. TSP: Tìm đường đi ngắn nhất trong đồ thị 17](#_Toc45305121)

[B. KNAPSAC: Bài toán cái túi ( n đồ vật có a cân và c giá trị) túi có b kg. Tìm giá trị max có thể mang 19](#_Toc45305122)

[C. BCA 21](#_Toc45305123)

[D. CBUS 24](#_Toc45305124)

[E. BCAP 26](#_Toc45305125)

[Train5: Quy hoạch động 28](#_Toc45305126)

[A. GOLD MINING 28](#_Toc45305127)

[J.WAREHOUSE (Giống bài A, thêm đk giằng buộc thời gian <=T) 30](#_Toc45305128)

[L. DRONE PICKUP 31](#_Toc45305129)

[F. NURSE (Số cách xếp các ngày nghỉ ở giữa sao cho các ngày đi làm [K1,K2]) 32](#_Toc45305130)

[Train 6: Đồ thị 34](#_Toc45305131)

[I. Lý thuyết Slide 34](#_Toc45305132)

[a) Tìm kiếm theo chiều sâu DFS - LIFO 34](#_Toc45305133)

[b) Tìm TPLT 34](#_Toc45305134)

[c) Cầu: Khi bỏ cạnh đó **đồ thị ko trọng số** mất tính liên thông 34](#_Toc45305135)

[d) Đồ thị có hướng => Tìm Thành phần liên thông mạnh 35](#_Toc45305136)

[e) Sắp xếp Topo: Công việc j cần hoàn thành trước công việc i 37](#_Toc45305137)

[f) Tìm kiếm theo chiều rộng – BFS: Thăm theo trình tự FIFO 37](#_Toc45305138)

[g) Đường đi ngắn nhất trên đồ thị không trọng số ( A->B qua ít cạnh nhất) 38](#_Toc45305139)

[Train 7: Đồ thị có trọng số 38](#_Toc45305140)

[I Lý thuyết slide 38](#_Toc45305141)

[a) Lưu đồ thị có trọng số 38](#_Toc45305142)

[b) Union Find : slide 39](#_Toc45305143)

[c) Tìm khung cây nhỏ nhất: Kruska + Prim 39](#_Toc45305144)

[d) Tìm đường đi ngắn nhất: 45](#_Toc45305145)

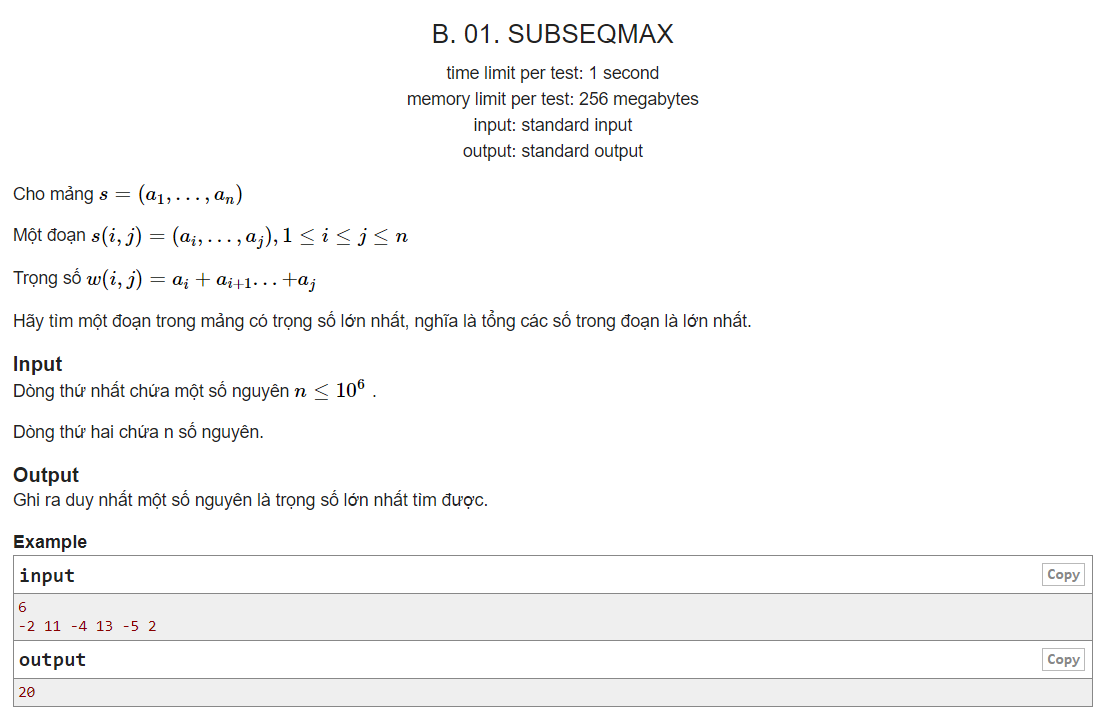
[e) Bài toán ghép cặp 45](#_Toc45305146)

[f) Bài toán tô màu đồ thị 45](#_Toc45305147)

[g) Kiểm tra đồ thị 2 phía 45](#_Toc45305148)

# Train 1

## B.01 Tìm đoạn có tổng lớn nhất

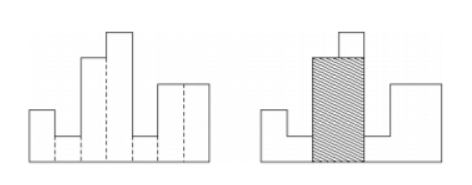


1. **#include** <bits/stdc++.h>

4. **using** **namespace** std;
6. **#define** all(s) s.begin(), s.end()
7. **#define** pb push\_back
8. **#define** ii pair<**int**, **int**>
9. **#define** x first
10. **#define** y second
11. **#define** bit(x, y) ((x >> y) & 1)
12. **#define** ull **unsigned** **long** **long**
13. **#define** ll **long** **long**
14. **int** n;
15. **int** A[1000000]; //(Khai báo mảng lớn => khai báo toàn cục)
17. **int** res;
18. **int** main() {
19. ios::sync\_with\_stdio(**false**); cin.tie(0);
20. cout.tie(0);
22. cin >> n;
23. memset(A,0,**sizeof**(A));
24. **for**(**int** i=0;i<n;i++){
25. cin >> A[i];
26. }
28. res=A[0];
30. **for**(**int** i=1;i<n;i++){
31. **if**(A[i-1]+A[i]>A[i]){
32. A[i]=A[i]+A[i-1];
33. }
34. **if**(A[i]>res) res=A[i];
35. }
37. cout << res;
38. **return** 0;
39. }

# Train2

## Hist – Tìm diện tích hình chữ nhật max



1. **#include** **<iostream>**
2. **#include<stdio.h>**
3. **#include** **<vector>**
4. **#include** **<stack>**
5. **using** **namespace** std;
7. **int** hist[1000000];
9. **long** **long** getMaxArea(**int** hist[], **int** n){
10. **stack**<**long** **long**> s;
11. **long** **long** max\_area = 0; **// Initialize max area**
12. **long** **long** tp; **// To store top of stack**
13. **long** **long** area\_with\_top; **// To store area with top bar**
14. **// as the smallest bar**
16. **// Run through all bars of given histogram**
17. **int** i = 0;
18. **while** (i < n)
19. {
20. **// If this bar is higher than the bar on top**
21. **// stack, push it to stack**
22. **if** (s.empty() || hist[s.top()] <= hist[i])
23. s.push(i++);
25. **// If this bar is lower than top of stack,**
26. **// then calculate area of rectangle with stack**
27. **// top as the smallest (or minimum height) bar.**
28. **// 'i' is 'right index' for the top and element**
29. **// before top in stack is 'left index'**
30. **else**
31. {
32. tp = s.top(); **// store the top index**
33. s.pop(); **// pop the top**
35. **// Calculate the area with hist[tp] stack**
36. **// as smallest bar**
37. area\_with\_top = hist[tp] \* (s.empty() ? i :
38. i - s.top() - 1);
40. **// update max area, if needed**
41. **if** (max\_area < area\_with\_top)
42. max\_area = area\_with\_top;
43. }
44. }
46. **// Now pop the remaining bars from stack and calculate**
47. **// area with every popped bar as the smallest bar**
48. **while** (s.empty() == **false**)
49. {
50. tp = s.top();
51. s.pop();
52. area\_with\_top = hist[tp] \* (s.empty() ? i :
53. i - s.top() - 1);
55. **if** (max\_area < area\_with\_top)
56. max\_area = area\_with\_top;
57. }
59. **return** max\_area;
60. }
61. **int** main()
62. {
63. **int** length;
64. cin >> length;
65. **while** (length != 0)
66. {
67. **for** (**int** i = 0; i < length; i++) {
68. cin >> hist[i];
69. }
71. cout << getMaxArea(hist, length) << **"\n"**;
73. cin >> length;
74. }

77. **//return 0;**

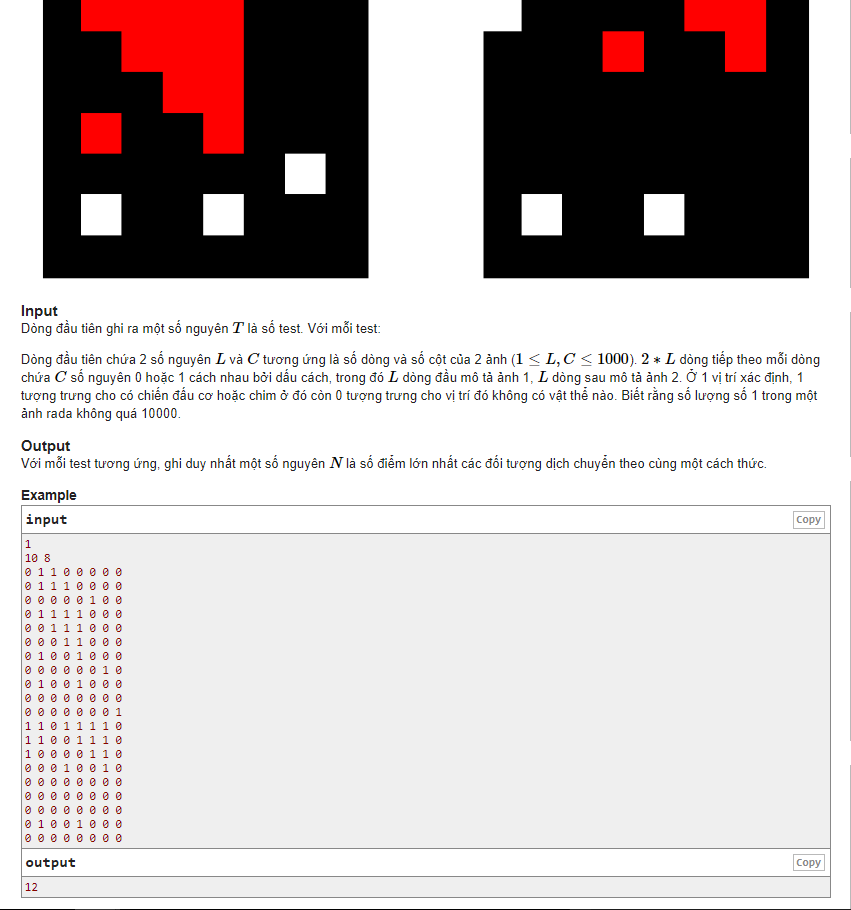
## Postman

Nhân viên đi từ điểm 0 ( mỗi lần đi chỉ đc giao <=k kiện) tới n khách hàng tại n địa điểm, mỗi khách có khoảng cách + số kiện cần giao. Tìm quãng đường giao min

1. **#include** <bits/stdc++.h>
2. **using** **namespace** std;
4. **#define** all(s) s.begin(), s.end()
5. **#define** ll **long** **long**
6. **#define** ull **unsigned** **long** **long**
7. **#define** ii pair<ll, ll>
9. **//int n;**
10. **//cin >> n;**
11. **//vector<int> a(n);**
12. **//for (int& e : a) cin >> e;**
13. **//sort(all(a)); //sx tang dan**
14. **//reverse(all(a)); //sx giam dan**
16. **using** **namespace** std;
18. **long** **long** n, k;
19. **vector<ii>** duongViTri;
20. **vector<ii>** amVitri;
22. **long** **long** **Time**(**vector<ii>** khoangCach) {
23. **long** **long** time = 0;
24. **for** (**int** i = 0; i < khoangCach.size(); i++) {
25. **if** (khoangCach[i].second == 0) **continue**;
26. time += (khoangCach[i].second / k) \* khoangCach[i].first \* 2;
27. khoangCach[i].second = khoangCach[i].second % k;
28. **if** (khoangCach[i].second == 0) **continue**;
29. **if** (i != khoangCach.size() - 1) {
30. time += khoangCach[i].first \* 2;
31. **long** **long** hangDu = k - khoangCach[i].second;
32. **long** **long** next = 1;
33. **while** (hangDu!=0 && (i+next) != khoangCach.size())
34. {
35. **if** (hangDu <= khoangCach[i + next].second) {
36. khoangCach[i + next].second -= hangDu;
37. hangDu = 0;
38. }
39. **else**
40. {
41. hangDu -= khoangCach[i + next].second;
42. khoangCach[i + next].second = 0;
43. }
44. next++;
45. }
46. }
47. **else**
48. {
49. time += 2 \* khoangCach[khoangCach.size() - 1].first;
50. }
51. }
52. **return** time;
53. }

56. **int** main()
57. {
58. cin >> n >> k;
59. **int** i, j;
60. **while** (n--)
61. {
62. cin >> i >> j;
63. **if** (i > 0) {
64. duongViTri.push\_back({ i,j });
65. }
66. **else**
67. {
68. amVitri.push\_back({ abs(i),j });
69. }
70. }
71. sort(all(duongViTri));
72. reverse(all(duongViTri));
73. **/\*for (ii& e : duongViTri) cout << e.first <<"-"<< e.second << " ";**
74. **cout << "\n";\*/**
75. sort(all(amVitri));
76. reverse(all(amVitri));
77. **/\*for (ii& e : amVitri) cout << e.first << "-" << e.second << " ";**
78. **cout << "\n";\*/**
79. cout << **Time**(duongViTri) + **Time**(amVitri);
80. **return** 0;
81. }

## Locate



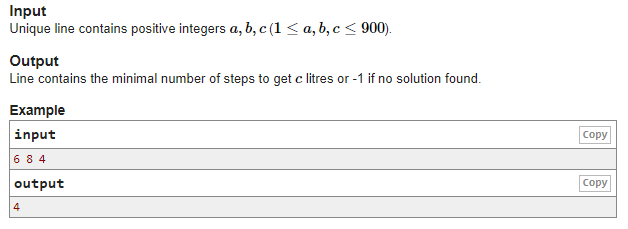
Cách 1:

1. **#include** **<list>**
2. **#include** **<locale>**
3. **#include** **<map>**
4. **#include** **<queue>**
5. **#include** **<set>**
6. **#include** **<vector>**
7. **#define** ii pair<**int**, **int**>
8. **using** **namespace** std;
10. **int** test;
11. **int** a[2001][2001];
12. **int** T, L, C, x;
13. **vector<ii>** v1;
14. **vector<ii>** v2;
16. **void** fill\_0\_a() {
17. **for** (**int** i = 0; i < 2001; i++) {
18. **for** (**int** j = 0; j < 2001; j++) {
19. a[i][j] = 0;
20. }
21. }
22. }
23. **int** main() {
24. cin >> test;
25. **while** (test--) {
26. **// 0<=x,y <= 1000; => min x1-x2 = -1000**
27. fill\_0\_a();
28. cin >> L >> C;
29. v1.clear();
30. v2.clear();
31. **for** (**int** i = 0; i < L; i++)
32. **for** (**int** j = 0; j < C; j++)
33. {
34. cin >> x;
35. **if** (x == 1)
36. v1.push\_back({ i,j });
37. }
38. **for** (**int** i = 0; i < L; i++)
39. **for** (**int** j = 0; j < C; j++)
40. {
41. cin >> x;
42. **if** (x == 1)
43. v2.push\_back({ i,j });
44. }
46. **int** res = 0;
47. **for** (**vector<ii>**::**iterator** it = v1.begin(); it != v1.end(); ++it)
48. **for** (**vector<ii>**::**iterator** jt = v2.begin(); jt != v2.end(); ++jt)
49. {
50. ii tmp1 = \*it;
51. ii tmp2 = \*jt;
53. **int** u = tmp1.first - tmp2.first+1000;
54. **int** v = tmp1.second - tmp2.second+1000;
55. a[u][v]++;
56. res = max(res, a[u][v]);
57. }
58. cout << res << endl;
59. }
60. **return** 0;
61. }

Cách 2: Dùng umap

1. **#define** toa\_do pair<**int**, **int**>
2. **#define** toa\_do\_pointer **vector<toa\_do>**::**iterator**
3. **using** **namespace** std;
5. **int** main() {
7. **int** test;
9. **int** T, L, C, x;
10. **vector<toa\_do>** v1;
11. **vector<toa\_do>** v2;
12. cin >> test;
13. **while** (test--) {
14. **map**<toa\_do, **int**> umap;
15. umap.clear();
16. cin >> L >> C;
17. v1.clear();
18. v2.clear();
19. **for** (**int** i = 0; i < L; i++)
20. **for** (**int** j = 0; j < C; j++)
21. {
22. cin >> x;
23. **if** (x == 1)
24. v1.push\_back({ i,j });
25. }
26. **for** (**int** i = L; i < 2\*L; i++)
27. **for** (**int** j = C; j < 2\*C; j++)
28. {
29. cin >> x;
30. **if** (x == 1)
31. v2.push\_back({ i-L,j-C });
32. }
33. **int** length = 0, res = 0;
35. **for** (toa\_do\_pointer i = v1.begin(); i != v1.end(); i++)
36. **for** (toa\_do\_pointer j = v2.begin(); j != v2.end(); j++)
37. {
38. toa\_do tmp1 = \*i;
39. toa\_do tmp2 = \*j;
40. umap[toa\_do(tmp1.first - tmp2.first, tmp1.second - tmp2.second)]++;
41. res = max(res, umap[toa\_do(tmp1.first - tmp2.first, tmp1.second - tmp2.second)]);
42. }
43. cout << res << endl;
44. }
45. **return** 0;
46. }

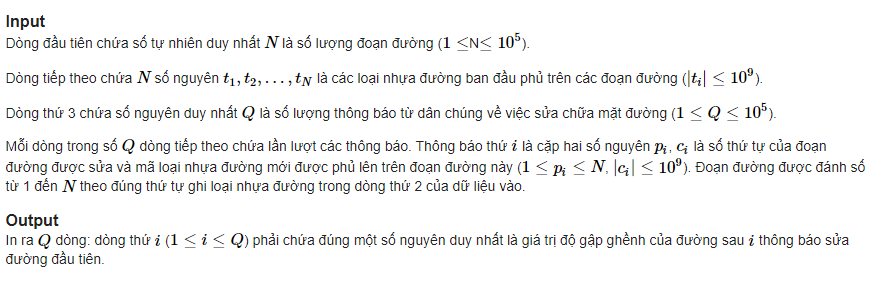
## WATERJUG BFS

Cho 2 chai: a lít và b lít. Cần bao nhiêu bước để lấy chính xác c lít

1. **#include<iostream>**
2. **#include<algorithm>**
3. **using** **namespace** std;
4. //Tim uoc chung lon nhat
5. **int** gcd(**int** a, **int** b) { //gcd(a,b) = gcd(b, a mod b) Thuật toán Euclid
6. **if**(b == 0)
7. **return** a;
8. **return** gcd(b, a % b);
9. }
11. **// a: pour from, b: pour to, c: destination**
12. **int** pour(**int** a, **int** b, **int** c) {
13. **int** from = a;
14. **int** to = 0;
15. **int** step = 1;
17. **while**((from != c) || (to != c)) {
18. **// find the smaller jug need to fill water**
19. **// fill that water with amount of temp water**
20. **int** temp = min(from, b - to);
22. to += temp;
23. from -= temp;
24. step++;
26. **if**((from == c) || (to == c))
27. **break**;
29. **// case 1: fisrt jug empty**
30. **if**(from == 0) {
31. from += a;
32. step++;
33. }
35. **// case 2: second jug empty**
36. **if**(to == b) {
37. to = 0;
38. step++;
39. }
40. }
41. **return** step;
42. }
44. **int** steps(**int** a, **int** b, **int** c) {
45. **if**(a > b)
46. swap(a, b);
48. **if**(c > b)
49. **return** -1;
51. **// c is multiple of gcd of a and b**
52. **if**(c % gcd(a,b) != 0)
53. **return** -1;
55. **// find min steps of two cases**
56. **return** min(pour(a, b, c), pour(b, a, c));
57. }
59. **int** main() {
60. **int** a, b, c;
61. cin >> a;
62. cin >> b;
63. cin >> c;
64. cout << steps(a, b, c);
65. **return** 0;
66. }

## REROAD

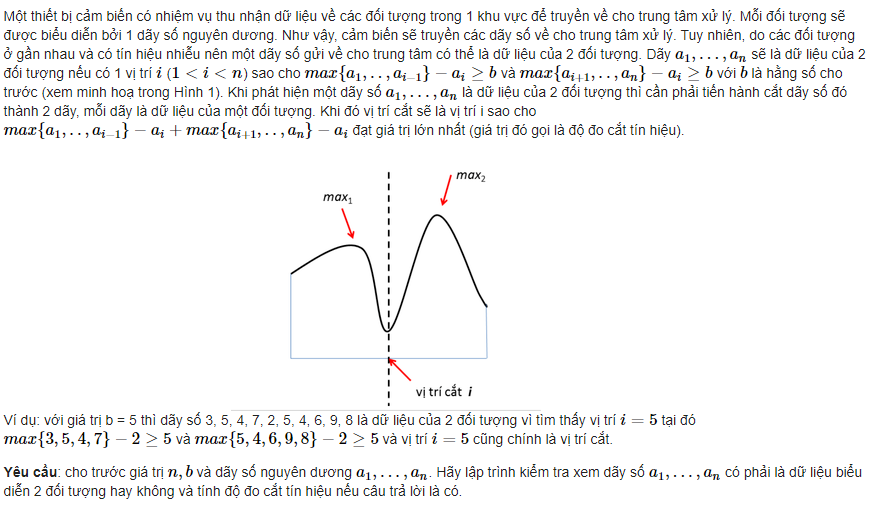
Tìm độ gập gềnh của đường

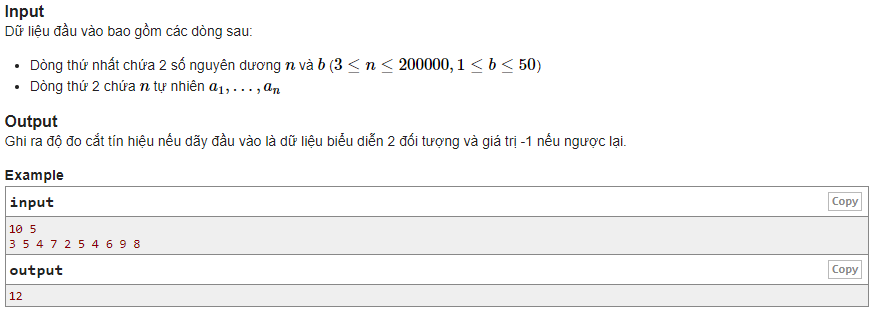




1. **#include** **<iostream>**
2. **using** **namespace** std;
3. **int** a[100000];
4. **/\*d[i] để tính có bao nhiêu độ gập gềnh**
5. **- =1 nếu a[i] != a[i-1]**
6. **- =0 còn lại**
7. **- Từ đó đếm số chỉ d[i] => Độ gập gềnh**
8. **- VD : 2 2 2 2 2 2 3 2 2**
9. **=> 1 0 0 0 (answer=1) 1 0 1 1 0 (answer=3)**
10. **\*/**
11. **int** d(**int** i) {
12. **if** (i == 1) **return** 1;
13. **return** a[i] != a[i - 1];
14. }
16. **int** main()
17. {
18. **int** N;
19. cin >> N;
20. **for** (**int** i = 1; i <= N; i++) {
21. cin >> a[i];
22. }
23. **int** answer = 0;
24. **for** (**int** i = 1; i <= N; i++) {
25. answer = answer + d(i);
26. }
27. **int** Q;
28. cin >> Q;
29. **int** p, c;
30. **while** (Q--)
31. {
32. cin >> p >> c;
33. **if** (p == N) {
34. answer = answer - d(p);
35. a[p] = c;
36. answer = answer + d(p);
37. }
38. **else** {
39. answer = answer - d(p + 1)-d(p);
40. a[p] = c;
41. answer = answer + d(p + 1)+d(p);
42. }
44. cout << answer << endl;
45. }
46. }

## SIGNAL





1. **#include** **<iostream>**
2. **using** **namespace** std;
4. **int** max\_prefix[200000];
5. **int** max\_suffix[200000];
6. **int** input[200000];
7. **int** max(**int** a, **int** b) {
8. **if** (a > b) {
9. **return** a;
10. }
11. **else**
12. {
13. **return** b;
14. }
15. }
16. **int** main()
17. {
18. **int** n, b;
19. cin >> n >> b;
20. **for** (**int** i = 0; i < n; i++)
21. {
22. cin >> input[i];
23. }
24. max\_prefix[0] = input[0];
25. max\_suffix[n-1] = input[n-1];
26. **for** (**int** i = 1; i < n; i++) {
27. max\_prefix[i] = max(max\_prefix[i - 1], input[i]);
28. }
29. **for** (**int** i = n - 2; i >= 0; i--) {
30. max\_suffix[i] = max(max\_suffix[i + 1], input[i]);
31. }
33. **int** res = -1;
34. **for** (**int** i = 0; i < n; i++) {
35. **int** tmp1 = max\_prefix[i - 1] - input[i];
36. **int** tmp2 = max\_suffix[i + 1] - input[i];
37. **if** (tmp1 >= b && tmp2 >= b) {
38. res = max(res, tmp1 + tmp2);
39. }
40. }
41. cout << res;
43. }

## Dễ

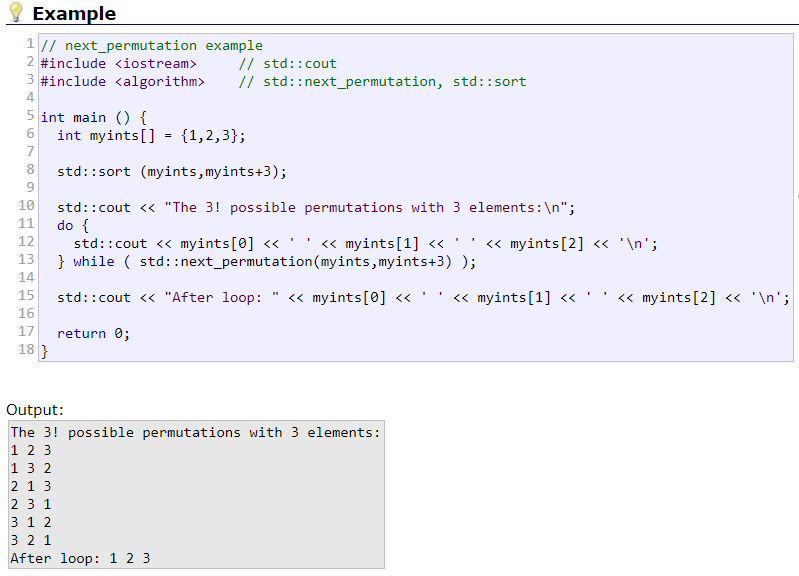
## PARENTHESES

Cho vào 1 xâu () [] {}. Kiểm tra xem nó có là 1 xâu hợp lệ

1. **#include** **<iostream>**
2. **using** **namespace** std;
3. **#include<stdio.h>**
4. **#include** **<stack>**
6. **// CPP program to check for balanced parenthesis.**

9. **// function to check if paranthesis are balanced**
10. **bool** areParanthesisBalanced(string expr)
11. {
12. **stack<char>** s;
13. **char** x;
15. **// Traversing the Expression**
16. **for** (**int** i = 0; i < expr.length(); i++)
17. {
18. **if** (expr[i] == **'('** || expr[i] == **'['** || expr[i] == **'{'**)
19. {
20. **// Push the element in the stack**
21. s.push(expr[i]);
22. **continue**;
23. }
25. **// IF current current character is not opening**
26. **// bracket, then it must be closing. So stack**
27. **// cannot be empty at this point.**
28. **if** (s.empty())
29. **return** **false**;
31. **switch** (expr[i])
32. {
33. **case** **')'**:
35. **// Store the top element in a**
36. x = s.top();
37. s.pop();
38. **if** (x == **'{'** || x == **'['**)
39. **return** **false**;
40. **break**;
42. **case** **'}'**:
44. **// Store the top element in b**
45. x = s.top();
46. s.pop();
47. **if** (x == **'('** || x == **'['**)
48. **return** **false**;
49. **break**;
51. **case** **']'**:
53. **// Store the top element in c**
54. x = s.top();
55. s.pop();
56. **if** (x == **'('** || x == **'{'**)
57. **return** **false**;
58. **break**;
59. }
60. }
62. **// Check Empty Stack**
63. **return** (s.empty());
64. }
66. **// Driver program to test above function**
67. **int** main()
68. {
69. **int** T;
70. cin >> T;
72. string expr;
73. **while** (T--)
74. {
75. cin >> expr;
76. **if** (areParanthesisBalanced(expr))
77. cout << **"1\n"**;
78. **else**
79. cout << **"0\n"**;
80. }
81. **return** 0;
82. }

# TRAIN 3



#include <bits/stdc++.h>

using namespace std;

#define all(s) s.begin(), s.end()

#define pb push\_back

#define ii pair<int, int>

#define x first

#define y second

#define bit(x, y) ((x >> y) & 1)

#define ll long long

//#define int long long

int N,K;

int A[11][11];

int stt[11];

int dem = 0;

bool Test() {

for (int i = N-1; i >= 0; i--) {

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

if (A[stt[i]][stt[j]] == 1) return false;

}

}

return true;

}

int main() {

ios::sync\_with\_stdio(false); cin.tie(0);

cout.tie(0);

memset(A, 0, sizeof(A));

memset(stt, 0, sizeof(stt));

cin >> N >> K;

while (K--) {

int i, j;

cin >> i >> j;

A[i][j] = 1;

}

for (int i = 0; i < N; i++) stt[i] = i+1;

do {

for (int i = 0; i < N; i++) cout<<stt[i]<<" ";

if (Test()) {

dem++;

}

cout << "\n";

} while (next\_permutation(stt, stt + N));

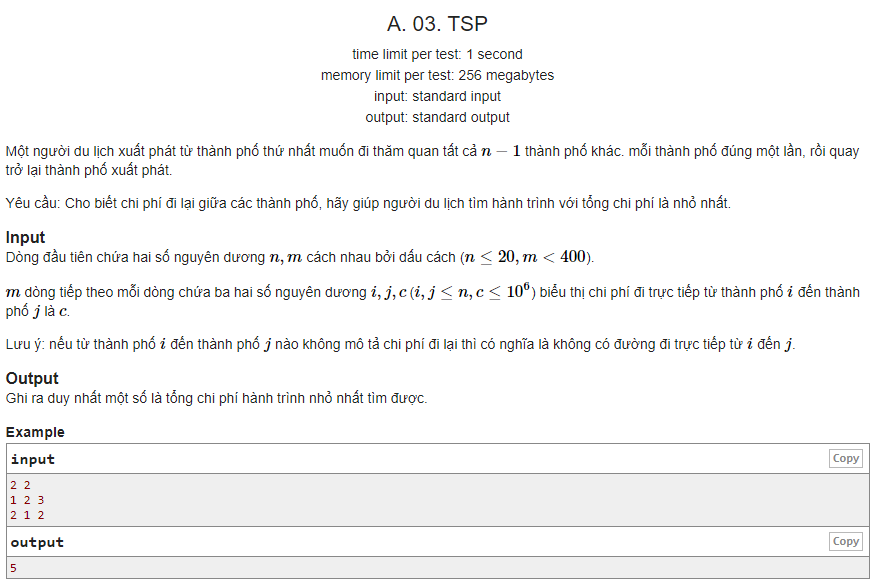
/\*4 3 4 2 2 1 3 2\*/

cout << dem;

return 0;

}

## TSP: Tìm đường đi ngắn nhất trong đồ thị



1. **using** **namespace** std;
2. **int** n, m;
3. **int**\*\* c;
4. **int**\* x; **//luu ds cac dinh di qua theo thu tu**
5. **int**\* mark;
6. **int** best = INT\_MAX;
7. **int** minc = INT\_MAX;
8. **//Cai dat bang de quy nhanh can**
10. **void** min\_C (){ //Tim trọng số cạnh nhỏ nhất
12. **for** (**int** i = 0; i <= n; i++) {
13. **for** (**int** j = 0; j <= n; j++) {
14. **if** (c[i][j] != 1e7 && c[i][j] < minc) minc = c[i][j];
15. }
16. }
18. }
19. **void** duyet(**int** i, **int** sum) {
20. **if** (i == n) {
21. **if** (sum + c[x[n]][1] < best) best = sum + c[x[n]][1];
22. **return**;
23. }
24. **for** (**int** j = 2; j <= n; j++) {
25. **if** (mark[j] == 0 && c[x[i]][j] != 1e7) {
26. **if** (sum + c[x[i]][j] +minc\*(n-i)<best) {
27. mark[j] = 1;
28. x[i + 1] = j;
29. duyet(i + 1, sum + c[x[i]][j]);
30. mark[j] = 0;
31. }
32. }
33. }
34. }
36. **int** main() {
37. cin >> n >> m;
38. x = (**int**\*)calloc(n + 1, **sizeof**(**int**));
39. mark = (**int**\*)calloc(n + 1, **sizeof**(**int**));
40. x[1] = 1;
41. mark[1] = 1;
43. c = (**int**\*\*)calloc(n + 1, **sizeof**(**int**\*));
44. **for** (**int** i = 0; i < n + 1; i++) c[i] = (**int**\*)calloc(n + 1, **sizeof**(**int**));
46. **for** (**int** i = 0; i <= n; i++) {
47. **for** (**int** j = 0; j <= n; j++) {
48. c[i][j]= 1e7;
49. }
50. }
51. **while** (m--)
52. {
53. **int** i, j, k;
54. cin >> i >> j >> k;
55. c[i][j] = k;
56. }
57. **//thanh pho: 1,2,3...n**
58. min\_C();
59. duyet(1, 0);
61. cout << best;

64. **return** 0;
65. }

int n, m;

int c[21][21];

/\*--------------------

Chỉ quan tâm đến đỉnh cuối được thăm và tập đỉnh trước nó

- save[X][p]: độ dài đường đi ngắn nhất từ tập đỉnh X và dừng tại p

- bit thứ i của X = 1 khi đỉnh i được thăm

\*/

int save[2097152][21];

int duyet(int X, int p) {

if (\_\_builtin\_popcount(X) == n) return c[p][0];

if (save[X][p] != 1e7) return save[X][p];

int ans = 2e9;

for (int s = 1; s < n; s++) {

if ((X >> s & 1) == 0) {

ans = min(ans, c[p][s] + duyet(1 << s | X, s));

}

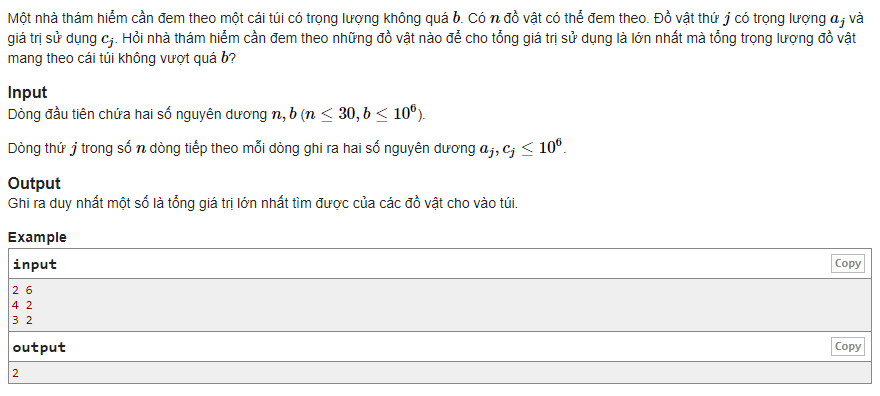
}

save[X][p] = ans;

return ans;

}

## KNAPSAC: Bài toán cái túi ( n đồ vật có a cân và c giá trị) túi có b kg. Tìm giá trị max có thể mang



Cách 1: (quy hoạch động)

1. **#include** **<vector>**
2. **#include** **<iostream>**
3. **#include** **<algorithm>**
4. **using** **namespace** std;
6. **int** n, W;
7. **int**\* w;
8. **int**\* p;
9. **int**\*\* res;
11. **void** **Fill\_res**() {
12. **//Fill chay dong 1 va 2: res[1]**
13. **//res[0][i] fill 0 res[i][0] fill 0**
14. **//res[1][i]**
15. **for** (**int** i = w[1]; i < W + 1; i++) res[1][i] = p[1];
17. **//res(i,j) = max{ res(i-1,j), res(i-1,j-w[i]) +p[i])**
18. **// i: 2-> i<n+1**
19. **// j: w[1] -> w< W+1**
20. **for** (**int** i = 2; i < n + 1; i++) {
21. **for** (**int** j = w[1]; j < W + 1; j++) {
22. **if** (j - w[i] < 0) res[i][j] = res[i - 1][j];
23. **if** (j - w[i] >= 0) {
24. res[i][j] = max(res[i - 1][j], res[i - 1][j - w[i]] + p[i]);
25. }
26. }
27. }
29. }


33. **int** main()
34. {
35. cin >> n >> W;
36. w = (**int**\*)calloc(n + 1, **sizeof**(**int**));
37. p = (**int**\*)calloc(n + 1, **sizeof**(**int**));
39. res = (**int**\*\*)calloc(n+1, **sizeof**(**int**\*));
40. **for** (**int** i = 0; i < n+1; i++) res[i] = (**int**\*)calloc(W+1, **sizeof**(**int**));
42. **/\* res:**
43. **0 1 2 3 .... W**
44. **w p 0**
45. **1**
46. **2**
47. **...**
48. **n**
49. **\*/**
51. **int** k = 1;
52. **int** tmp = n;
53. **while** (tmp--)
54. {
55. cin >> w[k] >> p[k];
56. k++;
57. }
59. **//-------------------------------------**
60. **//Sx tang dan w**
61. **for** (**int** i = 0; i < n; i++) {
62. **for** (**int** j = i+1; j < n + 1; j++) {
63. **if** (w[i] > w[j]) {
64. **int** tmp = w[i]; **int** tmp1 = p[i];
65. w[i] = w[j]; p[i] = p[j];
66. w[j] = tmp; p[j] = tmp1;
67. }
68. }
69. }
71. **Fill\_res**();
72. **int** kq = 0;
73. **for** (**int** i = 0; i < n + 1; i++) {
74. **for** (**int** j = 0; j < W + 1; j++) {
75. **if** (res[i][j] > kq ) kq = res[i][j];
76. }
77. }
78. cout << kq;
79. **return** 0;
80. }

Cách 2: Duyệt toàn bộ Bit 0/1 là chọn vật/ko chọn

int main()

{

int n, b;

cin >> n >> b;

int a[31];

int c[31];

memset(a, 0, sizeof(a));

memset(c, 0, sizeof(c));

for (int i = 0; i < n; i++) cin >> a[i] >> c[i];

int ans = 0;

for (int mask = 1 << n - 1; mask >= 0; mask--) {

int sumKhoiLuong = 0;

int sumGiaTri = 0;

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

if (mask >> i & 1 == 1) {

sumKhoiLuong += a[i];

sumGiaTri += c[i];

}

}

if (sumKhoiLuong <= b) ans = max(ans, sumGiaTri);

}

cout << ans;

return 0;

}

## BCA

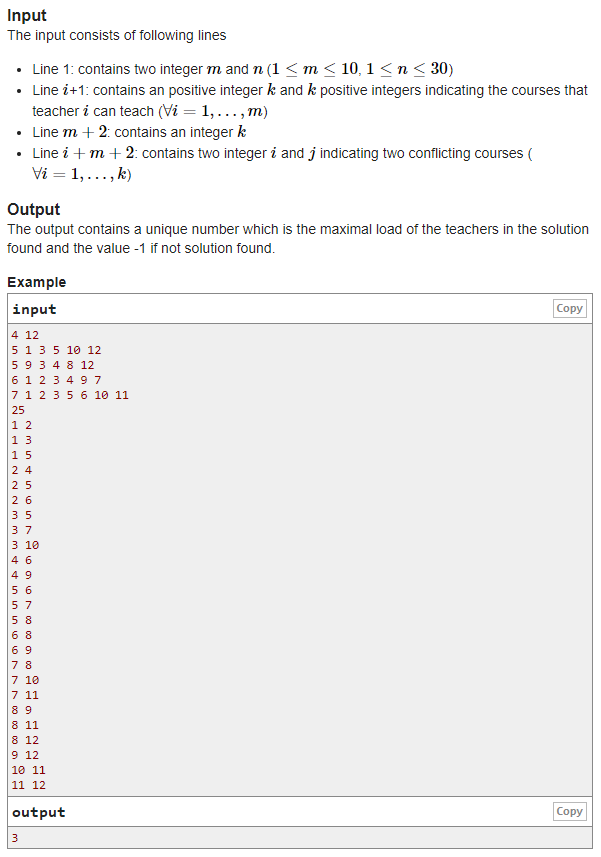
Có m giáo viên T={1,2,..m} và n khóa học C={1,2…,n}.

Mỗi giáo viên có 1 danh sách các khóa học có thể dạy được.

Có 1 danh sách các cặp đối ( ko thể dạy cùng bởi 1 giáo viên)

Tải trọn: Số khóa học giao cho người đó

* Tìm cách để tải trọng lớn nhất của các giáo viên là nhỏ nhất



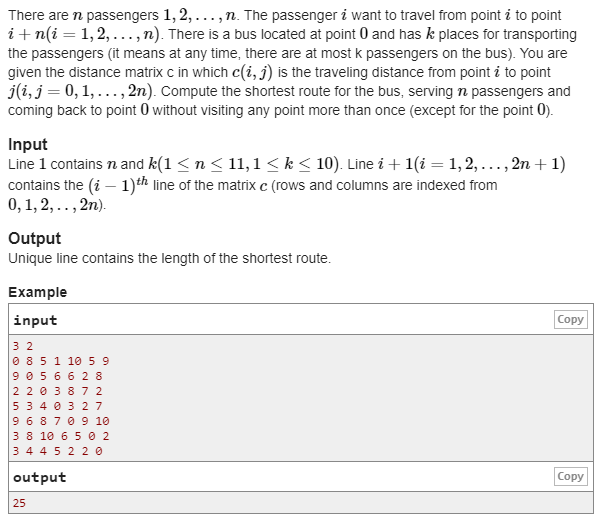
1. **#include** <bits/stdc++.h>
2. **using** **namespace** std;
4. **#define** MAX\_m 11 **//giao vien**
5. **#define** MAX\_n 31 **//khoa hoc**
6. **#define** ll **long** **long**
8. ll m, n;
9. **bool** **Class**[MAX\_m][MAX\_n] = { 0 }; **// giao vien nao co the day mon nao**
10. **bool** **Congv**[MAX\_m][MAX\_n] = { 0 }; **// giao vien dang day mon gi**
11. **bool** **Subj**[MAX\_n][MAX\_n] = { 0 }; **//mon hoc xung khắc**
12. ll **Load**[MAX\_m] = { 0 }; **// khoi luong cong viec**
13. ll ans = INT\_MAX;
14. ll a, max;
15. ll min;
17. **// Xem môn học mới có bị xung khắc với các môn học giáo viên đó đang dạy hay ko**
18. ll **Test**(ll i, ll k) {
19. **for** (**int** j = 1; j <= n; j++) {
20. **if** (**Congv**[i][j] == **true** && **Subj**[j][k] == **true**) **return** 0;
21. }
22. **return** 1;
23. }
24. **// Tìm tải nhiều nhất trong các giáo viên ở cách xếp hiện tại**
25. ll max\_CongViec() {
26. ll max = 0;
27. **for** (**int** i = 0; i < 11; i++) {
28. **if** (**Load**[i] > max) max = **Load**[i];
29. }
30. **return** max;
31. }
32. **//Thử xếp môn học k vào từng giáo viên**
33. **void** **Try**(ll k) {
34. **for** (**int** i = 1; i <= m; i++) { **//Thử từng người cho môn học k**
35. **if** (**Class**[i][k] == **true** && **Test**(i, k)) { **//Nếu người đó dạy đc môn k và**
36. **Congv**[i][k] = **true**;
37. **Load**[i]++;
38. **if** (k == n) {
39. ll max = max\_CongViec();
40. **if** (ans > max) ans = max;
41. }
42. **else** **if** (max\_CongViec() < ans) **Try**(k + 1);
43. **Load**[i]--;
44. **Congv**[i][k] = 0;
45. }
46. }
47. }
48. **int** main() {
49. cin >> m >> n;
50. ll k, g, j;
51. **//Nhập các môn giáo viên có thể dạy**
52. **for** (**int** i = 1; i <= m; i++) {
53. cin >> k;
54. **for** (j = 0; j < k; j++) {
55. cin >> g;
56. **Class**[i][g] = **true**;
57. }
58. }
59. **//Lập ra bảng các môn xung khắc**
60. cin >> k;
61. **for** (**int** i = 0; i < k; i++) {
62. cin >> g >> j;
63. **Subj**[g][j] = **true**;
64. **Subj**[j][g] = **true**;
65. }
66. **for** (**int** i = 0; i < 31; i++) {
67. **Subj**[i][i] = **true**;
68. }
70. **Try**(1); **//Thử sắp xếp môn học 1**
71. cout << ans;
72. **return** 0;
73. }

## CBUS

Có n hành khách (1,2,..n). Hành khách I muốn đi từ điểm I đến i+n

Một xe bus đi từ điểm 0 và có k chỗ ngồi. Cho ma trận khoảng cách c[i][j] (I,j=0,1,..,2n)

Tìm đường đi ngắn nhất phục vụ n hành khách và quay về điểm 0 (đi qua mỗi điểm 1 lần)



1. **#include** <bits/stdc++.h>
3. **using** **namespace** std;
4. **int** n, k, n1, minc; **//n1=2\*n+1**
5. **int**\*\* c;
6. **int** save[4194305][11];
7. **bool**\* flag = **new** **bool**[n1];
8. **void** fill\_c() {
9. c = (**int**\*\*)calloc(2 \* n + 1, **sizeof**(**int**\*));
10. **for** (**int** i = 0; i < 2 \* n + 1; i++) c[i] = (**int**\*)calloc(2 \* n + 1, **sizeof**(**int**));
12. **for** (**int** i = 0; i < 2 \* n + 1; i++) {
13. **for** (**int** j = 0; j < 2 \* n + 1; j++) {
14. cin >> c[i][j];
15. }
16. }
17. }
19. **void** min\_C() {
20. **for** (**int** i = 0; i < n1; i++) {
21. **for** (**int** j = 0; j < n1; j++) {
22. **if** (c[i][j] != 0 && c[i][j] < minc) minc = c[i][j];
23. }
24. }
26. }
28. **/\***
29. **duong di: 0-> 1,2-n...2n ->0**
30. **Tap cac dinh X: 0(0) 0(1) 0(2) 0(3) 0(4) 0(5) 0(6)**
32. **0 8 5 1 10 5 9**
33. **9 0 5 6 6 2 8**
34. **2 2 0 3 8 7 2**
35. **5 3 4 0 3 2 7**
36. **9 6 8 7 0 9 10**
37. **3 8 10 6 5 0 2**
38. **3 4 4 5 2 2 0**
39. **\*/**
40. **int** tong = 0;
41. **int** duyet(**int** X, **int** p) {
42. **int** i = \_\_builtin\_popcount(X);
43. **if** (i == n1) **return** c[p][0];
44. **if** (X < 4194305) {
45. **if** (save[X][p] != -1) **return** save[X][p];
46. }
47. **int** ans = 2e9;
49. **for** (**int** s = 1; s < n1; s++) {
50. **if** ((X >> s & 1) == 0 && c[p][s] + minc \* (n1 - i) < ans) { **//Nếu s chưa được thăm và nhánh cận**
51. **if** (s <= n) { **//Nếu s là điểm đón khách**
52. **if** (tong + 1 > k) **continue**; **//Nếu xe đầy khách thì đến điểm khác**
53. flag[s] = **true**;
54. tong++;
55. }
56. **else** { **//Nếu s là điểm trả khách**
57. **if** (flag[s - n] == **false**) **continue**; **//TH đến điểm trả mà chưa đón khách => thử s mới**
58. tong--;
59. }
61. ans = min(ans, c[p][s] + duyet(1 << s | X, s));
63. **if** (s <= n) {
64. tong--;
65. flag[s] = **false**;
66. }
67. **else** {
68. tong++;
69. }
70. }
71. }
72. **if** (X < 4194305) {
73. save[X][p] = ans;
74. }
75. **return** ans;
76. }

79. **int** main() {
80. cin >> n >> k;
81. minc = 1e9;
82. fill\_c();
83. n1 = 2 \* n + 1;
84. memset(flag, **false**, **sizeof**(flag));
85. memset(save, -1, **sizeof**(save));
87. min\_C();
88. **int** res = duyet(1, 0);
90. cout << res;
91. **return** 0;
92. }

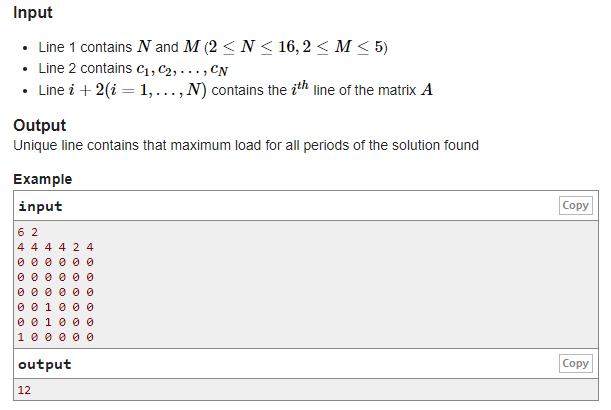
## BCAP

Có N khóa học 1,2,…,N cần phải đăng ký vào M kì học 1,2,…M

Mỗi khóa học I có trọng số c

Cho A[i][j] =1 => môn I cần học trước môn j

* Tìm cách chia để tải lớn nhất trong các kì là nhỏ nhất



* + Duyệt toàn bộ. Mỗi môn học chọn cho nó 1 kì học. Khi chọn xong hết. Kiểm tra điều kiện

1. **#include** <bits/stdc++.h>
2. **using** **namespace** std;
4. **int** N, M;
5. **int** c[17];
6. **int** A[17][17];
8. **int** termCources[17];
9. **int** loadTerm[6];
10. **int** ans=INT\_MAX;
12. **int** **MaxLoad**() {
13. **int** max = 0;
14. **for** (**int** i = 1; i <= M; i++) {
15. **if** (loadTerm[i] > max)max = loadTerm[i];
16. }
17. **return** max;
18. }
19. **int** **Test**() { **//Kiểm tra TH có TM ma trận A**
20. **for** (**int** i = 1; i <= N; i++) {
21. **for** (**int** j = 1; j <= N; j++) {
22. **if** (A[i][j] == 1) {
23. **if** (termCources[i] > termCources[j]) **return** 0;
24. }
25. }
26. }
27. **return** 1;
28. }
29. **//Chọn kì học cho môn học**
30. **void** **Try**(**int** k) {
31. **for** (**int** i = 1; i <= M; i++) {
32. termCources[k] = i;
33. loadTerm[i]+=c[k];
35. **if** (k == N) {
36. **//for (int i = 1; i <= N; i++) cout << termCources[i] << " ";**
37. **//cout << "\n";**
38. **if** (**Test**()) {
39. **int** max = **MaxLoad**();
40. **if** (max < ans) ans = max;
41. }
42. }
43. **else** **Try**(k + 1);
45. loadTerm[i]-=c[k];
46. termCources[k] = 0;
47. }
48. }
50. **int** main() {
51. memset(c, 0, **sizeof**(c));
52. memset(A, 0, **sizeof**(A));
53. memset(termCources, 0, **sizeof**(termCources));
54. memset(loadTerm, 0, **sizeof**(loadTerm));
55. cin >> N >> M;
56. **for** (**int** i = 1; i <= N; i++) {
57. cin >> c[i];
58. }
59. **for** (**int** i = 1; i <= N; i++) {
60. **for** (**int** j = 1; j <= N; j++) {
61. cin >> A[i][j];
62. }
63. }
64. **Try**(1);
65. cout << ans;
66. **return** 0;
67. }

# Train5: Quy hoạch động

## Lý thuyết slide

## Dãy Fibonacci

int mem[1000];

for (int i = 0; i < 1000; i++)

mem[i] = -1;

int fibonacci(int n) {

if (n <= 2) {

return 1;

}

if (mem[n] != -1) {

return mem[n];

}

int res = fibonacci(n - 2) + fibonacci(n - 1);

mem[n] = res;

return res;

}

## Dãy con tăng dài nhất ( dãy con = Mảng ban đầu, loại bỏ đi 1 số phần tử)

* List[i]: dãy con tăng dài nhất mảng a[0]->a[i] mà kết thúc tại i

int a[1000];

int mem[1000];

memset(mem, -1, sizeof(mem));

int lis(int i) {

if (mem[i] != -1) {

return mem[i];

}

int res = 1;

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

if (a[j] < a[i]) {

res = max(res, 1 + lis(j));

}

}

mem[i] = res;

return res;

}

* Tìm ra max trong mem[i]

int mx = 0;

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

mem[i]=list(i);

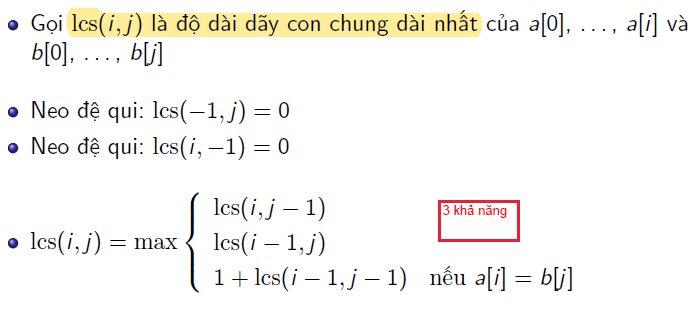
mx = max(mx , mem[i]);

}

## Dãy con chung dài nhất

a ="bananinn"

b ="kaninn" => Độ dài: 5



string a = " bananinn ",

b = " kaninan ";

int mem[1000][1000];

memset(mem, -1, sizeof(mem));

int lcs(int i, int j) {

if (i == -1 || j == -1) {

return 0;

}

if (mem[i][j] != -1) {

return mem[i][j];

}

int res = 0;

res = max(res, lcs(i, j - 1));

res = max(res, lcs(i - 1, j));

if (a[i] == b[j]) {

res = max(res, 1 + lcs(i - 1, j - 1));

}

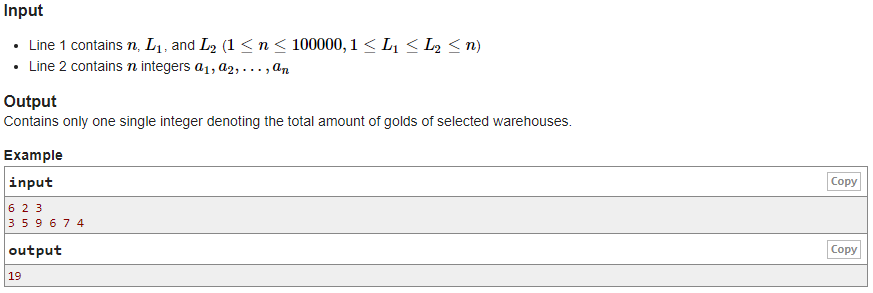
mem[i][j] = res;

return res;

}

## GOLD MINING

N căn nhà chứa vàng (1,2..,N). Căn nhà I có lượng vàng ai. Tìm tổng lượng vàng max biết 2 căn nhà có khoảng cách nằm trong [L1,L2]



* + QHĐ 1 chiều:

1. **#include** <bits/stdc++.h>
2. **//A. 05. GOLD MINING**
4. **//Quy hoach vong lap**
5. **using** **namespace** std;
7. **#define** all(s) s.begin(), s.end()
8. **#define** pb push\_back
9. **#define** ii pair<**int**, **int**>

**#define** x first

**#define** y second

**#define** bit(x, y) ((x >> y) & 1)

2. **int** n, L1, L2;
3. **int** a[100000];
4. **int** mem[100000]; **//mem[i]so vang lon nhat ket thuc tại a[i]**
6. **//L1<= distance <= L2**

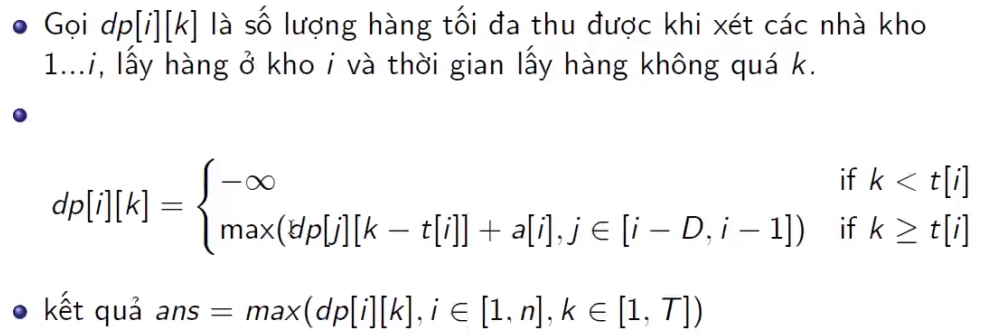
9. **int** main() {
10. ios::sync\_with\_stdio(**false**); cin.tie(0);
11. cout.tie(0);
12. cin >> n >> L1 >> L2;
13. memset(a, 0, 100000);
14. memset(mem, 0, 100000);
16. **for** (**int** i = 0; i < n; i++) {
17. cin >> a[i];
18. mem[i] = a[i];
19. }
21. **for** (**int** i = L1; i < n; i++) {
22. **for** (**int** j = i - L2; j <= i - L1; j++) {
23. **if** (j >= 0) {
24. **if** (mem[i] < mem[j] + a[i]) mem[i] = mem[j] + a[i];
25. }
26. **else** {
27. **continue**;
28. }
29. }
30. }
31. **int** res = mem[0];
32. **for** (**int** i = 0; i < n; i++) {
33. res = max(res, mem[i]);
34. }
35. cout << res;
36. **return** 0;
37. }

## J.WAREHOUSE (Giống bài A, thêm đk giằng buộc thời gian <=T)

Chiếc xe tải lên kế hoạch tới N trạm (1,2,..N). Mỗi trạm có a (hàng) và t(thời gian để lấy hàng)

Khoảng cách giữa 2 trạm [0,D] và thời gian lấy hàng <= T

* Tìm đường để tổng hàng đc chọn là max



1. **#include** <bits/stdc++.h>
2. **using** **namespace** std;
4. **#define** ll **long** **long**
5. **#define** ull **unsigned** **long** **long**
6. **#define** MAXN 1001
7. **#define** MAXK 401
9. **int** N, T, D;
10. **int** a[MAXN];
11. **int** b[MAXN];
12. **int** sumGoods[MAXN][MAXK]; //sumGoods[i][j] Số hàng tối đa thu được khi xét các nhà
13. //kho 1,..,I lấy hàng ở kho I và thời gian lấy ko quá k
14. **int** ans;
16. **int** main()
17. {
18. cin >> N >> T >> D;
20. **for** (**int** i = 1; i <= N; i++) {
21. cin >> a[i];
22. }
24. **for** (**int** i = 1; i <= N; i++) {
25. cin >> b[i];
26. }


30. ans = a[1];
31. sumGoods[1][b[1]] = a[1];
33. **for** (**int** i = 1; i <= N; i++) {
34. **for** (**int** j = max(1, i - D); j < i; j++) {
35. **for** (**int** k = 0; k <= T - b[i]; k++) {
36. sumGoods[i][k + b[i]] = max(sumGoods[i][k + b[i]], sumGoods[j][k] + a[i]);
37. ans = max(ans, sumGoods[i][k + b[i]]);
38. }
39. }

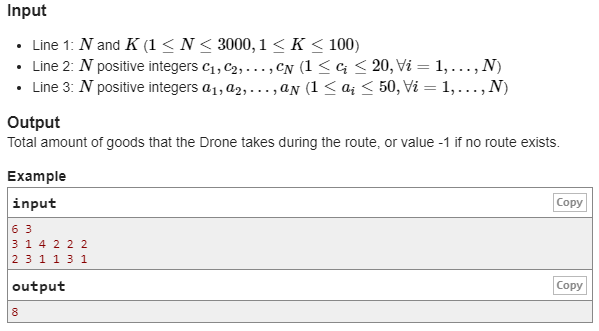
42. }
44. cout << ans;
45. }

## L. DRONE PICKUP

N địa điểm 1,2,..,N. Mỗi địa điểm có c lượng hàng, a năng lượng

Một máy bay cần bay từ 1->N:

* + Không được dừng quá K+1 điểm ( kể cả điềm đầu cuối)
  + Dừng ở I thì đi xa nhất i+ai.
* Tìm lộ trình để lấy nhiều hàng nhất



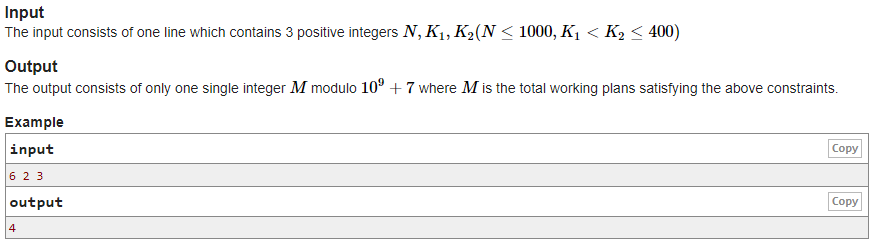
1. **using** **namespace** std;
3. **int** N, K;
4. **int** c[3001]; **//hàng**
5. **int** a[3001]; **//năng lượng**
6. **int** dp[3001][102]; **//Hàng tối đa thu được khi đi từ 1,..,i lấy hàng ở i và số điểm lấy hàng <= k+1**

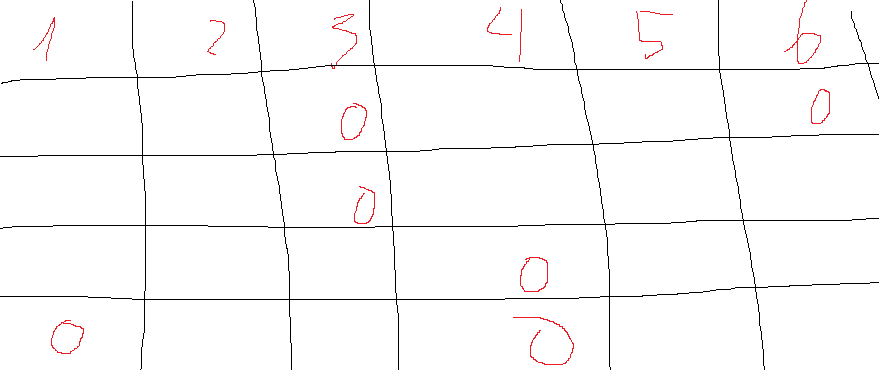
9. **int** main() {
11. cin >> N >> K;
12. memset(c, 0, **sizeof**(c));
13. memset(a, 0, **sizeof**(a));
15. **for** (**int** i = 1; i <= N; i++) {
16. cin >> c[i];
17. }
18. **int** D = 0;
19. **for** (**int** i = 1; i <= N; i++) {
20. cin >> a[i];
21. **if** (a[i] > D) D = a[i];
22. }
23. dp[1][1] = c[1];
25. **for** (**int** i = 1; i < N; i++) {
26. **for** (**int** k = 1; k <= K; k++) {
27. **if** (dp[i][k] != 0) {
28. **for** (**int** j = 1; j <= a[i]; j++) {
29. **if** (i + j <= N) dp[i + j][k + 1] = max(dp[i + j][k + 1], dp[i][k] + c[i + j]);
30. }
31. }
32. }
33. }
34. **int** ans = dp[N][1];
35. **for** (**int** i = 1; i <= K + 1; i++) {
36. **if** (dp[N][i] > ans) ans = dp[N][i];
37. }
39. cout << ans;
40. }

## NURSE (Số cách xếp các ngày nghỉ ở giữa sao cho các ngày đi làm [K1,K2])

Có N ngày 1,2,..,N. Y tá có thể làm việc trong [K1,K2] xong nghỉ một ngày rồi làm tiếp.

Tìm số cách có thể sắp xếp.





1. **#include** <bits/stdc++.h>
3. **using** **namespace** std;
5. **int** n, k1, k2;
6. **int** ans[10];

9. **int** main()
10. {
11. cin >> n >> k1 >> k2;
13. ans[0]=ans[1] = 1;
15. **for** (**int** i = 0; i <= n + 1; i++) {
16. **if** (ans[i] != 0) {
17. **for** (**int** j = k1 + 1; j <= k2 + 1; j++) {
18. **if** (i + j <= n + 1) {
19. ans[i + j] += ans[i];
20. ans[i + j] = ans[i + j] % 100000007;
21. }
22. }
23. }
24. }
25. **//for (int i = 0; i <= n + 1; i++) cout << ans[i]<<" ";**
26. **//cout << "\n";**
27. **int** tmp = ans[n] + ans[n + 1];
28. tmp %= 100000007;
29. cout << tmp;
30. }

# Train 6: Đồ thị

## Lý thuyết Slide

### Tìm kiếm theo chiều sâu DFS - LIFO

vector <int > adj[1000];

vector <bool > visited(1000, false);

void dfs(int u) {

if (visited[u])

return;

visited[u] = true;

for (int i = 0; i < adj[u].size(); i++) {

int v = adj[u][i];

dfs(v);

}

}

### Tìm TPLT

vector <int > adj[1000];

vector <int > component(1000, -1);

void find\_component(int cur\_comp, int u) {

if (component[u] != -1)

return;

component[u] = cur\_comp;

for (int i = 0; i < adj[u].size(); i++) {

int v = adj[u][i];

find\_component(cur\_comp, v);

}

}

int main()

{

int n=6;

adj[0].push\_back(1);

adj[0].push\_back(2);

adj[1].push\_back(0);

adj[1].push\_back(2);

adj[2].push\_back(0);

adj[2].push\_back(1);

adj[2].push\_back(3);

adj[3].push\_back(2);

adj[4].push\_back(5);

adj[5].push\_back(4);

int components = 0; //STT TPLT

for (int u = 0; u < n; u++) {

if (component[u] == -1) {

find\_component(components, u);

components++;

}

}

for (int i = 0; i < n; i++) cout << component[i] << " ";

}

### Cầu: Khi bỏ cạnh đó **đồ thị ko trọng số** mất tính liên thông

#define ii pair<int, int>

const int n = 1000;

vector <int > adj[n], low(n), num(n, -1);

int curnum = 0;

vector <ii> bridges;

void find\_bridges(int u, int p) {

low[u] = num[u] = curnum++;

for (int i = 0; i < adj[u].size(); i++) {

int v = adj[u][i];

if (v == p) continue;

if (num[v] == -1) {

find\_bridges(v, u);

low[u] = min(low[u], low[v]);

}

else {

low[u] = min(low[u], num[v]);

}

if (low[v] > num[u]) {

bridges.push\_back(make\_pair(u, v));

}

}

}

int main()

{

int n=6;

adj[0].push\_back(1);

adj[0].push\_back(2);

adj[1].push\_back(0);

adj[1].push\_back(2);

adj[2].push\_back(0);

adj[2].push\_back(1);

adj[2].push\_back(3);

adj[3].push\_back(2);

adj[4].push\_back(5);

adj[5].push\_back(4);

for (int u = 0; u < n; u++) {

if (num[u] == -1)

find\_bridges(u, -1);

}

for (ii &e: bridges)cout << e.first << e.second << " ";

}

### Đồ thị có hướng => Tìm Thành phần liên thông mạnh

vector <int > adj[100];

vector <int > low(100), num(100, -1);

vector <bool > connect(100, false);

int curnum = 0;

stack <int > comp;

void scc(int u) {

comp.push(u);

connect[u] = true;

low[u] = num[u] = curnum++;

for (int i = 0; i < adj[u].size(); i++) {

int v = adj[u][i];

if (num[v] == -1) {

scc(v);

low[u] = min(low[u], low[v]);

}

else if (connect[v]) {

low[u] = min(low[u], num[v]);

}

}

if (num[u] == low[u]) {

printf(" TPLT : ");

while (true) {

int cur = comp.top();

comp.pop();

connect[cur] = false;

printf("%d , ", cur);

if (cur == u) break;

}

printf("\n");

}

}

int main()

{

int n=4;

adj[0].push\_back(1);

adj[1].push\_back(2);

adj[2].push\_back(0);

adj[2].push\_back(1);

adj[2].push\_back(3);

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

if (num[i] == -1) {

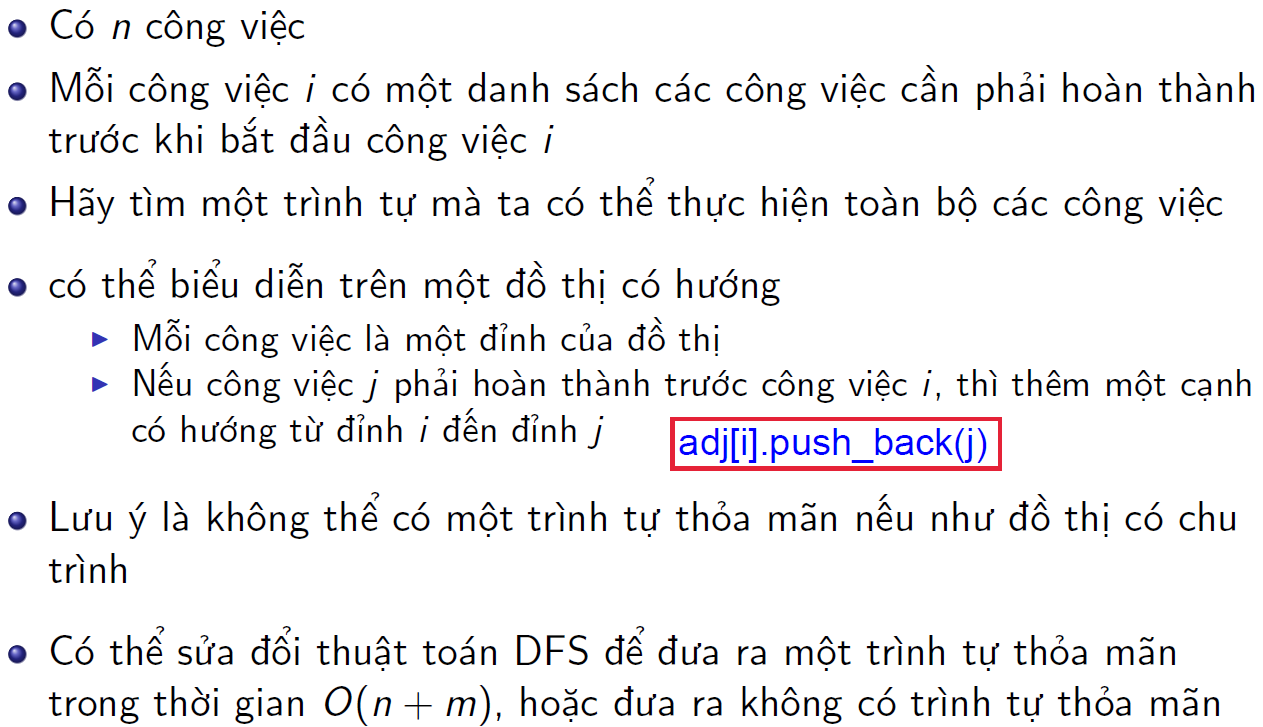
scc(i);

}

}

}

### Sắp xếp Topo: Công việc j cần hoàn thành trước công việc i



vector <int > adj[1000];

vector <bool > visited(1000, false);

vector <int > order;

void topsort(int u) {

if (visited[u])

return;

visited[u] = true;

for (int i = 0; i < adj[u].size(); i++) {

int v = adj[u][i];

topsort(v);

}

order.push\_back(u);

}

Void main(){

for (int u = 0; u < n; u++)

topsort(u);

}

### Tìm kiếm theo chiều rộng – BFS: Thăm theo trình tự FIFO

vector <int > adj[1000];

vector <bool > visited(1000, false);

queue <int > Q;

Q.push(start);

visited[start] = true;

while (!Q.empty()) {

int u = Q.front(); Q.pop();

for (int i = 0; i < adj[u].size(); i++) {

int v = adj[u][i];

if (!visited[v]) {

Q.push(v);

visited[v] = true;

}

}

}

### Đường đi ngắn nhất trên đồ thị không trọng số ( A->B qua ít cạnh nhất)

* Gọi BFS từ A đến khi gọi được B

vector <int > adj[1000];

vector <int > dist(1000, -1);

queue <int > Q;

Q.push(A);

dist[A] = 0;

while (!Q.empty()) {

int u = Q.front(); Q.pop();

for (int i = 0; i < adj[u].size(); i++) {

int v = adj[u][i];

if (dist[v] == -1) {

Q.push(v);

dist[v] = 1 + dist[u];

}

}

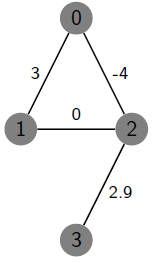
}

printf("%d\n", dist[B]);

# Train 7: Đồ thị có trọng số

## I Lý thuyết slide

## Lưu đồ thị có trọng số



struct edge {

int u, v;

int weight;

edge(int \_u, int \_v, int \_w) {

u = \_u;

v = \_v;

weight = \_w;

}

};

int main()

{

vector <edge > adj[4];

adj[0].push\_back(edge(0, 1, 3));

adj[0].push\_back(edge(0, 2, -4));

adj[1].push\_back(edge(1, 0, 3));

adj[1].push\_back(edge(1, 2, 0));

adj[2].push\_back(edge(2, 0, -4));

adj[2].push\_back(edge(2, 1, 0));

adj[2].push\_back(edge(2, 3, 2.9));

adj[3].push\_back(edge(3, 2, 2.9));

}

## Union Find : slide

## Tìm khung cây nhỏ nhất: Kruska + Prim

* + Thuật toán Kruska

//Kruska dựa trên geek

/\*bài tập: https://onlinejudge.org/external/116/p11631.pdf

Đây mới xử lý phần tìm khung cây nhỏ nhất

Cần thêm:

- Tìm ra các ngã 3

- Cho đèn tại các ngã 3 đó sáng

- Tìm khung cây

\*/

#include <bits/stdc++.h>

using namespace std;

#define ii pair<int, int>

int n, m;

//--------------GEEKSFORGEEKS-----------------------------------

typedef pair<int, int> iPair;

// Structure to represent a graph

struct Graph

{

int V, E;

vector< pair<int, iPair> > edges;

// Constructor

Graph(int V, int E)

{

this->V = V;

this->E = E;

}

// Utility function to add an edge

void addEdge(int u, int v, int w)

{

edges.push\_back({ w, {u, v} });

}

// Function to find MST using Kruskal's

// MST algorithm

int kruskalMST();

};

// To represent Disjoint Sets

struct DisjointSets

{

int\* parent, \* rnk;

int n;

// Constructor.

DisjointSets(int n)

{

// Allocate memory

this->n = n;

parent = new int[n + 1];

rnk = new int[n + 1];

// Initially, all vertices are in

// different sets and have rank 0.

for (int i = 0; i <= n; i++)

{

rnk[i] = 0;

//every element is parent of itself

parent[i] = i;

}

}

// Find the parent of a node 'u'

// Path Compression

int find(int u)

{

/\* Make the parent of the nodes in the path

from u--> parent[u] point to parent[u] \*/

if (u != parent[u])

parent[u] = find(parent[u]);

return parent[u];

}

// Union by rank

void merge(int x, int y)

{

x = find(x), y = find(y);

/\* Make tree with smaller height

a subtree of the other tree \*/

if (rnk[x] > rnk[y])

parent[y] = x;

else // If rnk[x] <= rnk[y]

parent[x] = y;

if (rnk[x] == rnk[y])

rnk[y]++;

}

};

/\* Functions returns weight of the MST\*/

int Graph::kruskalMST()

{

int mst\_wt = 0; // Initialize result

// Sort edges in increasing order on basis of cost

sort(edges.begin(), edges.end());

// Create disjoint sets

DisjointSets ds(V);

// Iterate through all sorted edges

vector< pair<int, iPair> >::iterator it;

for (it = edges.begin(); it != edges.end(); it++)

{

int u = it->second.first;

int v = it->second.second;

int set\_u = ds.find(u);

int set\_v = ds.find(v);

// Check if the selected edge is creating

// a cycle or not (Cycle is created if u

// and v belong to same set)

if (set\_u != set\_v)

{

// Current edge will be in the MST

// so print it

cout << u << " - " << v << endl;

// Update MST weight

mst\_wt += it->first;

// Merge two sets

ds.merge(set\_u, set\_v);

}

}

return mst\_wt;

}

//--------------------------------------------------------------

int main()

{

//--------------GEEKSFORGEEKS------------------------------

cin >> n >> m;

Graph g(n, m);

int i, j, k;

while (m--)

{

cin >> i >> j >> k;

g.addEdge(i, j, k);

}

cout << "Edges of MST are \n";

int mst\_wt = g.kruskalMST();

cout << "\nWeight of MST is " << mst\_wt;

return 0;

/\*vector <edge > ans;

ans= mst(n);

int sum = 0;

for (edge& e : edges) {

cout << e.u << "-" << e.v << ": " << e.weight << "\n";

sum += e.weight;

}

cout << sum;\*/

}

/\*Sample Input

7 11

0 1 7

0 3 5

1 2 8

1 3 9

1 4 7

2 4 5

3 4 15

3 5 6

4 5 8

4 6 9

5 6 11

0 0

Sample Output

51\*/

//Code slide

/\*bài tập: https://onlinejudge.org/external/116/p11631.pdf

Đây mới xử lý phần tìm khung cây nhỏ nhất

Cần thêm:

- Tìm ra các ngã 3

- Cho đèn tại các ngã 3 đó sáng

- Tìm khung caay nhỏ nhất

\*/

#include <bits/stdc++.h>

using namespace std;

#define ii pair<int, int>

struct edge {

int u, v;

int weight;

edge(int \_u, int \_v, int \_w) {

u = \_u;

v = \_v;

weight = \_w;

}

};

vector <int > parent;

struct union\_find {

union\_find(int n) {

parent = vector <int >(n);

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

parent[i] = i;

}

}

int find(int x) { //Trả về phần tử tập đại diện chứ x

if (parent[x] == x) {

return x;

}

else {

parent[x] = find(parent[x]); // nen parent

return parent[x];

}

}

void unite(int x, int y) { //Trộn tập chứa x và tập chứa y

parent[find(x)] = find(y);

}

};

int n, m; //Số đỉnh, số cạnh

//Kruskal tìm khung cây nhỏ nhất

vector <edge> edges; //lưu danh sách cạnh

bool edge\_cmp(const edge& a, const edge& b) {

return a.weight < b.weight;

}

vector <edge > mst\_Kruskal(int n) {

union\_find uf(n);

sort(edges.begin(), edges.end(), edge\_cmp);

vector <edge > res;

for (int i = 0; i < edges.size(); i++) {

int u = edges[i].u,

v = edges[i].v;

if (uf.find(u) != uf.find(v)) {

uf.unite(u, v);

res.push\_back(edges[i]);

}

}

return res;

}

//Prim tìm khung cây nhỏ nhất

vector <vector <edge >> adj; //lưu danh sách cạnh theo đỉnh

vector <edge > mst\_Prim(int n) {

priority\_queue < pair <int, int >, vector <pair <int, int >>,

greater <pair <int, int >> > pq;

vector <edge > res;// edges set of mst

vector <int > bestW(n + 1, 1e9), bestAdj(n + 1, -1);

bestW[1] = 0;

pq.push({ bestW[1] , 1 });

while (res.size() < n - 1) {

while (!pq.empty() &&

pq.top().first != bestW[pq.top().second]) pq.pop();

if (pq.empty()) return res;// Given graph is not connected

int w = pq.top().first, v = pq.top().second, u = bestAdj[v];

for (edge e : adj[v])

if (bestW[e.v] > e.weight) {

bestW[e.v] = e.weight;

bestAdj[e.v] = e.u;

pq.push({ bestW[e.v], e.v });

}

if (v != 1) res.push\_back({ u , v , w });

}

return res;

}

int main()

{

cin >> n >> m;

int i, j, k;

while (m--)

{

cin >> i >> j >> k;

edges.push\_back(edge(i, j, k));

//adj[i].push\_back(edge(i, j, k));

}

vector <edge > ans1,ans2;

ans1 = mst\_Kruskal(n);

int sum = 0;

for (edge& e : ans1) {

cout << e.u << "-" << e.v << ": " << e.weight << "\n";

sum += e.weight;

}

cout << sum<<"\n" << "\n" << "\n";

return 0;

}

/\*7 11

0 1 7

0 3 5

1 2 8

1 3 9

1 4 7

2 4 5

3 4 15

3 5 6

4 5 8

4 6 9

5 6 11

\*/

## Tìm đường đi ngắn nhất:

Từ đỉnh xuất phát đến tất cả các đỉnh còn lại

- Dijkstra – Cho trọng số ko âm

- Bellman-Ford – Cho trọng số âm

## Bài toán ghép cặp

## Bài toán tô màu đồ thị

Tô màu sao cho 2 đình kề nhau khác màu

## Kiểm tra đồ thị 2 phía

//Kiểm tra đồ thị 2 phía

vector <int > adj[1000];

vector <int > side(1000, -1);

bool is\_bipartite = true;

void check\_bipartite(int u) {

for (int i = 0; i < adj[u].size(); i++) {

int v = adj[u][i];

if (side[v] == -1) {

side[v] = 1 - side[u];

check\_bipartite(v);

}

else if (side[u] == side[v]) {

is\_bipartite = false;

}

}

}

for (int u = 0; u < n; u++) {

if (side[u] == -1) {

side[u] = 0;

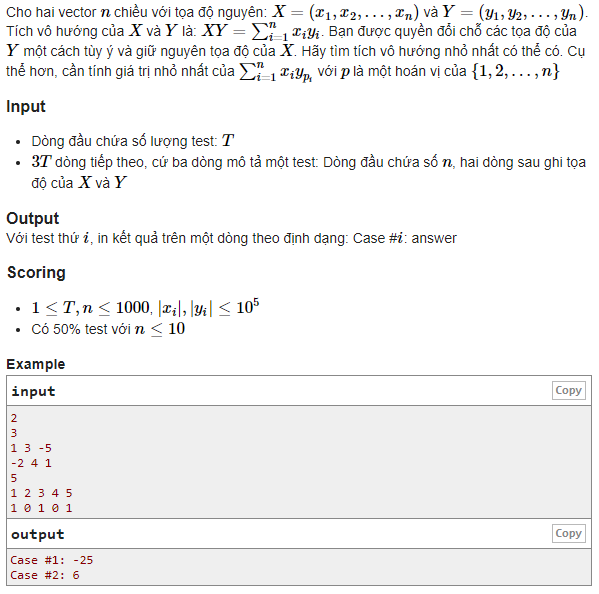
check\_bipartite(u);

}

}

# Demo Contest

## Scalar



#include <bits/stdc++.h>

using namespace std;

#define all(s) s.begin(), s.end()

#define pb push\_back

#define ii pair<int, int>

#define x first

#define y second

#define bit(x, y) ((x >> y) & 1)

int main() {

ios::sync\_with\_stdio(false); cin.tie(0);

cout.tie(0);

int ntest;

cin >> ntest;

for (int test = 1; test <= ntest; test++) {

int n;

cin >> n;

vector<int> a(n);

vector<int> b(n);

for (int& e : a) cin >> e;

for (int& e : b) cin >> e;

sort(all(a));

sort(all(b));

reverse(all(b));

long long ans = 0;

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

ans += 1LL \* a[i] \* b[i];

}

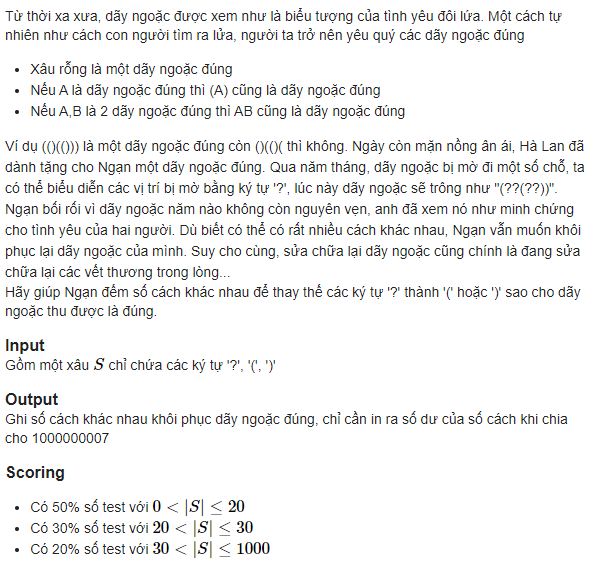
cout << "Case #" << test << ": " << ans << '\n';

}

return 0;

}

## BRCOUNT



#include <bits/stdc++.h>

#define MOD 1000000007

using namespace std;

#define all(s) s.begin(), s.end()

#define pb push\_back

#define ii pair<int, int>

#define x first

#define y second

#define bit(x, y) ((x >> y) & 1)

string s;

stack<char> stk;

int restore;

void TRY(int k) {

if (k == s.length()) {

if (stk.empty()) {

restore++; //đến vị trí cuối mà ko còn '('

return;

}

if (s[k] == '(') {

stk.push(s[k]);

TRY(k + 1);

stk.pop();

return;

}

if (s[k] == ')') {

if (stk.empty()) return;

char top = stk.top();

if (top == '(') {

stk.pop();

TRY(k + 1);

stk.push(top);

}

return;

}

// if s[k] == '?'

stk.push('(');

TRY(k + 1);

stk.pop();

if (stk.empty()) return;

char top = stk.top();

if (top == '(') {

stk.pop();

TRY(k + 1);

stk.push(top);

}

}

}

int main() {

ios::sync\_with\_stdio(false); cin.tie(0);

cout.tie(0);

cin >> s;

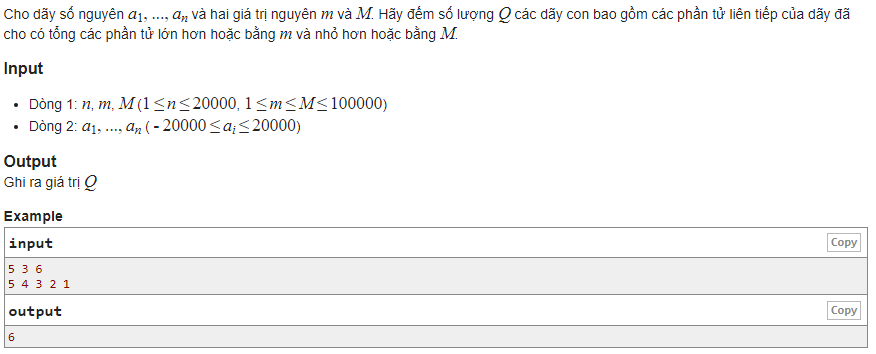
TRY(0);

cout << restore % MOD << endl;

return 0;

}

## BOUND SUBSEQ



#include <bits/stdc++.h>

using namespace std;

#define all(s) s.begin(), s.end()

#define pb push\_back

#define ii pair<int, int>

#define x first

#define y second

#define bit(x, y) ((x >> y) & 1)

int n, m, M;

int main() {

ios::sync\_with\_stdio(false); cin.tie(0);

cout.tie(0);

cin >> n >> m >> M;

vector<int> a(n);

for (int& e : a) cin >> e;

//sort(a.begin(), a.end());

int ans = 0;

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

int sum = 0;

for (int j = i; j < n; j++) {

sum += a[j];

if (sum >= m && sum <= M) ans++;

}

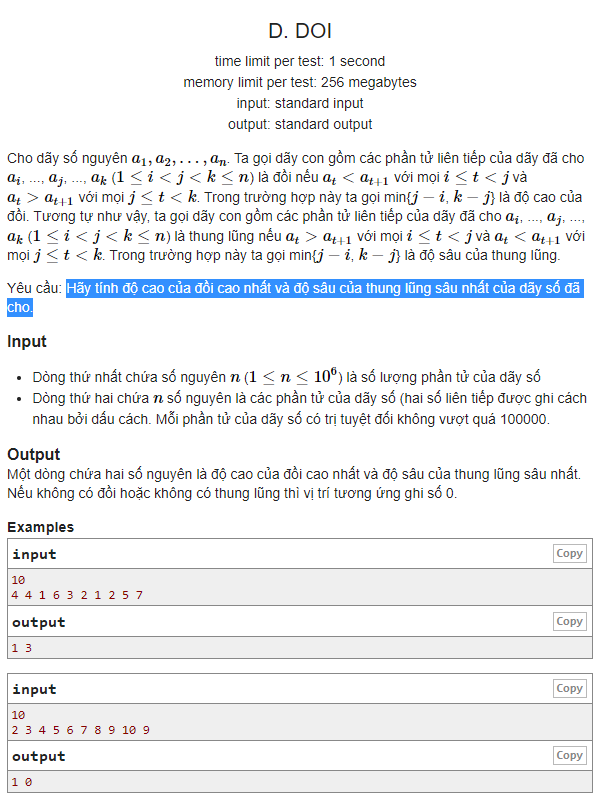
}

cout << ans;

return 0;

}

## DOI



#include <bits/stdc++.h>

using namespace std;

#define all(s) s.begin(), s.end()

#define pb push\_back

#define ii pair<int, int>

#define x first

#define y second

#define bit(x, y) ((x >> y) & 1)

#define ll long long

int n, a[1000001];

void solve() {

int len = 0;

int xuong = 0;

int ans = 0;

for (int i = 0; i < n-1; i++) {

if (a[i] < a[i + 1]) {

if (xuong != 0) { //TH dang xuong nui mà lên núi => kết thúc núi

len = 0;

xuong = 0;

}

len++; //Tính bđ núi mới

}

if (a[i] > a[i + 1]) {

xuong++;

}

ans = max(ans, min(len, xuong));

}

cout << ans;

}

int main() {

ios::sync\_with\_stdio(false); cin.tie(0);

cout.tie(0);

memset(a, 0, sizeof(a));

cin >> n;

for (int i = 0; i < n; i++) cin >> a[i];

solve();

for (int i = 0; i < n; i++) a[i] = -a[i];

solve();

return 0;

}

## PASSWORD mid term

Bình và Tân thuê 1 căn hộ. Họ quyết định tạo một mật khẩu chung để mở khóa:

* Bình viết ra dãy X gồm n số nguyên dương