SỞ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG THPT CHUYÊN**

**THUẬT TOÁN ĐƯỜNG ĐI NGẮN NHẤT DIJKSTRA**

Người viết chuyên đề:

Bộ môn: **Tin học**

Đơn vị: **Trường THPT Chuyên**

Mã số: …………………..

# Phần thứ nhất

# MỞ ĐẦU

1. **Lý do chọn đề tài**

Đề thi học sinh quốc gia, các kì thi Online, …trong các năm gần đây bài tập về đồ thị luôn chiếm một tỉ lệ lớn. Các bài toán ngày càng nâng cao độ khó về thuật toán và cấu trúc dữ liệu. Tuy nhiên để học sinh có thể giải hoặc giải một phần các bài toán đồ thị trong đề thi HSG QG, cần phải có cơ sở lý thuyết và kĩ năng chắc chắn. Chuyên đề này hướng tới các bạn học sinh Khối 10 khi học đồ thị, nhằm mục đích trang bị cho học sinh hiểu biết và kĩ năng về thuật toán tìm đường đi ngắn nhất.

Trong đề tài tôi trình bày các bài tập áp dụng thuật toán **Dijkstra,** Đây là thuật toán quen thuộc với các Thầy Cô nên tôi không trình bày chi tiết lý thuyết lại nữa, mục đích của chuyên đề là hệ thống bài tập cơ bản để phục vụ cho các em rèn luyện.

1. **Mục địch nghiên cứu**

Mục đích tôi viết chuyên đề này vừa là để tổng hợp lại những kiến thức, sưu tầm lại hệ thống bài tập phục vụ cho công tác dạy đội tuyển.

1. **Thời gian địa điểm**

Đề tại được tôi thực hiện trong năm học 2020 - 2021 đối với lớp 10 Chuyên Tin tại trường THPT Chuyên.

1. **Đòng góp mới về mặt thực tiễn**

Các bài toán trong các đề thi HSGQG luôn được cập nhật một cách liên tục và có phổ kiến thức rất rộng. Vì vậy, ngoài việc trang bị cho học sinh những kiến thức nền cơ bản thì cần rèn luyện cho học sinh kĩ năng phân loại, đoán nhận thuật toán. Để học sinh có thể dễ dạng hơn trong việc đoán nhận dạng toán sử dụng thuật toán Dijkstra.

Các bài tập được giới thiệu trong bài viết này đều được lựa chọn kĩ lưỡng có đủ cả test chấm, lời giải và chương trình kèm theo. Vì vậy, chuyên đề sẽ là một tài liệu phục vụ thiết thực cho việc giảng dạy đội tuyển ôn thi học sinh giỏi quốc gia, khu vực, các kì thi online,....

Kết quả việc thực hiện chuyên đề: học sinh nắm bắt được các bước cài đặt thuật toán, hứng thú và tích cực làm bài tập.

# Phần thứ hai

# NỘI DUNG CHUYÊN ĐỀ

**I. Thuật toán** **Dijkstra**

Cho đồ thị có hướng gồm đỉnh và cạnh, mỗi cạnh có một độ dài nhất định. Với mỗi đỉnh của đồ thị xác định độ dài của đường đi ngắn nhất từ đỉnh 1 tới đỉnh .

**DỮ LIỆU**

* Dòng đầu tiên chứa số nguyên ;
* dòng tiếp theo, mỗi dòng chứa 3 số nguyên mô tả 1 cạnh đi từ tới độ dài .

**KẾT QUẢ**

* Gồm 1 dòng chứa số nguyên trong đó là độ dài đường đi ngắn nhất từ đỉnh 1 tới đỉnh . Nếu không có đường đi tới đỉnh thì ghi .

**VÍ DỤ**

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| 5 5  1 2 5  1 3 2  3 4 1  1 4 6  3 5 5 | 5 2 3 7 |

**CÁCH TIÉP CẬN THUẬT TOÁN**

* Thuật toán tìm đường đi ngắn nhất Dijkstra là một thuật toán tham lam.
* Tương tự như thuật toán BFS tìm thành phần liên thông trên đồ thị, thuật toán Dijkstra cũng dựa trên tư tưởng “loang”, nhưng là “loang” kết quả tốt nhất
* Do đó việc cài đặt thuật toán Dijkstra cũng có sự tương đồng đối với BFS.

**CỤ THỂ CODE**

|  |
| --- |
| #include <stdio.h>  #include <vector>  #include <queue>  using namespace std;  const int oo = 1000111000;  typedef pair<int, int> ii;  vector<ii> a[2309];  int n, m;  int d[2309];  void dijkstra() {  priority\_queue<ii, vector<ii>, greater<ii>> pq;  for (int i = 1; i <= n; i++)  d[i] = oo;  d[1] = 0;  pq.push(ii(0, 1));  while (pq.size()) {  int u = pq.top().second;  int du = pq.top().first;  pq.pop();  if (du != d[u])  continue;  for (int i = 0; i < a[u].size(); i++) {  int v = a[u][i].second;  int uv = a[u][i].first;  if (d[v] > du + uv) {  d[v] = du + uv;  pq.push(ii(d[v], v));  }  }  }  }  int main() {  int p, q, m, w;  scanf("%d%d", &n, &m);  while (m--) {  scanf("%d%d%d", &p, &q, &w);  a[p].push\_back(ii(w, q));  a[q].push\_back(ii(w, p));  }  dijkstra();  for (int i = 1; i <= n; i++)  printf("d( 1 -> %d ) = %d\n", i, d[i]);  } |

**II. Bài tập áp dụng**

Thông qua các bài toán cơ bản sẽ giúp các em ghi nhớ lý thuyết và dần có kĩ năng cài đặt tốt hơn. Đây là nền tảng giúp các em có thể giải được các bài toán ở mức độ cao hơn.

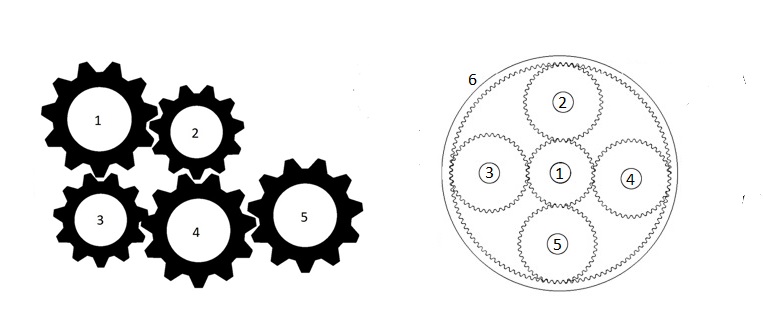
**BÀI 1. Bánh xe**

**Gears.cpp / Gears.inp / Gears.out / 1s / 1024 Mb**

Alice và Bob đang học về cơ khí và họ gặp đôi chút khó khăn về sự chuyển động của các bánh răng. Việc kết nối phức tạp và họ cần bạn giúp đỡ.

Một hệ thống các bánh răng bao gồm các bánh răng trong đó một số cặp liên kết chuyển động với nhau. Về mặt cơ học, hai bánh răng kết nối với nhau tức là các bánh răng của chúng cài vào nhau và khi bánh răng này chuyển động kéo theo chuyển động của bánh răng kia theo chiều ngược lại.

Mỗi bánh răng thuộc một trong 2 loại. Bánh răng là bánh răng loại kết nối ngoài nếu kết nối chuyển động với một bánh răng khác thì bánh răng đó nằm ngoài . Bánh răng là bánh răng loại kết nối trong nếu kết nối chuyển động với một bánh răng khác thì bánh răng đó nằm trong .



Trong ví dụ bên trái có 5 bánh răng đánh số từ 1 tới 5, có 5 sự kết nối giữa các bánh răng là Nếu ta quay bánh răng 1 theo dương, kéo theo bánh răng 2 và 3 quay theo chiều âm, kéo theo bánh răng 4 quay theo chiều dương. Cuối cùng bánh răng 5 quay theo chiều âm.

Trong ví dụ bên phải có 6 bánh răng, có sự kết nối giữ các bánh răng là Khi quay bánh răng 1 theo chiều dương, các bánh răng 2, 3, 4, 5 quay theo chiều âm, bánh răng 6 quay theo chiều âm.

Cho biết hệ thống liên kết giữa các bánh răng và truy vấn, mỗi truy vấn Alice và Bob mỗi người chọn một bánh răng (có thể trùng nhau) và quay theo chiều nhất định. Hỏi hệ thống bánh răng có ổn thỏa không? Hệ thống ổn thỏa nếu không có hai bánh răng nào bị tác động lực dẫn quay theo hai chiều khác nhau?

**DỮ LIỆU**

* Dòng đầu tiên chứa 3 số nguyên là số bánh răng của hệ thống, số cặp liên kết giữa các bánh răng và số truy vấn.
* Dòng thứ 2 chứa số nguyên trong đó nếu bánh răng thứ nếu liên kết với một bánh răng nào đó thì bánh răng đó nằm bên ngoài bánh răng , , nếu bánh răng thứ nếu liên kết với một bánh răng nào đó thì bánh răng đó nằm bên trong bánh răng .
* dòng tiếp theo mỗi dòng chứa hai số nguyên phân biệt mô tả có liên kết chuyển động giữa hai bánh răng .
* dòng tiếp theo, mỗi dòng chứa 4 số nguyên cho biết bánh răng mà Alice chọn, bánh răng mà Bob chọn, chiều chuyển động của bánh răng chiều chuyển động của bánh răng .

**KẾT QUẢ**

Gồm dòng, dòng thứ ghi YES nếu hệ thống ổn thỏa, ghi NO nếu hệ thống không ổn thỏa.

**VÍ DỤ**

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| 8 8 3  1 1 1 1 1 1 1 1  1 2  1 3  2 4  3 4  4 5  6 7  7 8  8 6  1 5 1 1  1 5 1 -1  6 7 1 1 | NO  YES  NO |

**THUẬT TOÁN**

* Trong mỗi thành phần liên thông, ta sẽ chọn một bánh xe làm gốc và cho quay bánh xe này theo chiều dương, sau đó lan truyền chuyển động sang các bánh xe khác trong cùng thành phần liên thông
* Với mỗi truy vấn ta kiểm tra tra xem hai bánh xe có cùng thành phần liên thông hay không, nếu cùng thành phần liên thông thì chúng cùng hay ngược chiều quay.
* Độ phức tạp .
* Mặc dù bài toán này chưa phải là bài toán về đường đi ngắn nhất nhưng sẽ giúp học sinh có hình dung về thuật toán loang giống như lan truyền chuyển động bánh xe, còn thuật toán **Dijkstra** cũng là lan truyền, nhưng là lan truyền kết quả tốt nhất. Qua đó có tính kế thừa giữa thuật toán Loang và thuật toán **Dijkstra.**

**CODE**

|  |
| --- |
| #include <bits/stdc++.h>  using namespace std;  #define nmax 100009  int n, m, qq;  int a[nmax];  vector <int> adj[nmax];  int g1, g2, d1, d2;  int d[nmax];  bool ok[nmax], cd[nmax];  int t, tp[nmax];  void dfs (int u)  {  ok[u] = 0;  tp[u] = t;  for (int i=0; i<adj[u].size(); i++)  {  int v= adj[u][i];  if (ok[v])  {  d[v] = a[u]\*a[v]\* d[u]\*(-1);  dfs(v);  }  else  {  if (d[v]\*d[u]\*a[u]\*a[v]==1)  cd[t] = 0;  }  }  }      int main()  {  //freopen("1.i", "r", stdin);  scanf("%d", &n);  scanf("%d", &m);  scanf("%d", &qq);  for (int i=1; i<=n; i++)  scanf("%d", &a[i]);  for (int i=1; i<=m; i++)  {  int x, y;  scanf("%d%d", &x, &y);  adj[x].push\_back(y);  adj[y].push\_back(x);  }  memset(ok, 1, sizeof(ok));  memset(cd, 1, sizeof(cd));  for (int i=1; i<=n; i++)  {  if (ok[i])  {  t++;  d[i]=1;  dfs(i);  }  }  while (qq--)  {  scanf("%d", &g1);  scanf("%d", &g2);  scanf("%d", &d1);  scanf("%d", &d2);  int t1 = tp[g1];  int t2 = tp[g2];  if (cd[t1]==0)  {  printf("NO\n");  continue;  }  if (cd[t2]==0)  {  printf("NO\n");  continue;  }  if (t1!=t2)  {  printf("YES\n");  continue;  }    int tg = d1\*d2\*d[g1]\*d[g2];  if (tg==1) printf("YES\n");  else printf("NO\n");  }  return 0;  } |

**BÀI 2. Đường đi ngắn nhất có điều kiện**

**ShortestDK.cpp / ShortestDK.inp / ShortestDK.out / 2s / 1024 Mb**

Có thành phố và con đường hai chiều. Cần vận chuyển các mặt hàng thiết yếu từ thành phố 1 tới tất cả các thành phố khác (việc vận chuyển tới các thành phố khác luôn thực hiện được).

Nhưng mọi con đường đều có thu phí. May mắn thay, bạn có thẻ ưu đãi, có nghĩa là khi đi từ thành phố 1 tới bất kì thành phố nào khác, bạn có thể chọn tối đa con đường và không bị tính phí khi đi qua con đường đó.

Bạn cần xác định chi phí tối thiểu để có thể chuyển hàng tới mỗi thành phố. Lưu ý bạn có thể xem việc vận chuyển hàng hóa từ thành phố 1 tới các thành phố khác là độc lập nhau.

**DỮ LIỆU**

* Dòng đầu tiên chứa số nguyên ;
* dòng tiếp theo, mỗi dòng chứa 3 số nguyên mô tả 1 con đường nối 2 thành phố với chi phí để đi qua.

**KẾT QUẢ**

* Gồm số nguyên, số thứ ứng với chi phí tối thiểu với thành phố .

**VÍ DỤ**

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| 5 6 1  1 2 2  1 3 6  2 4 6  2 5 8  3 5 4  4 5 1 | 0 0 0 2 2 |

**THUẬT TOÁN**

* Đây là bài toán đường đi ngắn nhất có điều kiện
* Do đó ta nghĩ tới ý tưởng là gọi là chi phí nhỏ nhất đi từ đỉnh 1 tới mà đã sử dụng thẻ ưu đãi
* Khi xét cạnh từ ta cập nhật min cho và
* Độ phức tạp là.
* Cảm nhận: Bài này có hướng kết hợp giữa quy hoạch động và thuật toán đường đi ngắn nhất.

**CODE**

|  |
| --- |
| #include <bits/stdc++.h>  using namespace std;  typedef int ll;  #define int long long  const int maxn=5e5+5;  vector >graph[maxn];  int cost[maxn][20];  bool vis[maxn][20];  int n,m,k;  void djkstra() {  priority\_queue > >pq;  pq.push(make\_pair(0, make\_pair(1, 0)));  cost[1][0]=0;  while (!pq.empty()) {  pair >curr=pq.top();  pq.pop();  int curr\_n=curr.second.first;  int curr\_s=curr.second.second;  int cst=cost[curr\_n][curr\_s];  for (int i=0; ic\_cst) {  cost[graph[curr\_n][i].first][curr\_s]=c\_cst;  pq.push(make\_pair(c\_cst\*(-1), make\_pair(graph[curr\_n][i].first, curr\_s)));  }  c\_cst=cst;  if(curr\_s==k) continue;  if(cost[graph[curr\_n][i].first][curr\_s+1]==-1) {  pq.push(make\_pair(c\_cst\*(-1), make\_pair(graph[curr\_n][i].first, curr\_s+1)));  cost[graph[curr\_n][i].first][curr\_s+1]=c\_cst;  }  else if(cost[graph[curr\_n][i].first][curr\_s+1]>c\_cst) {  cost[graph[curr\_n][i].first][curr\_s+1]=c\_cst;  pq.push(make\_pair(c\_cst\*(-1), make\_pair(graph[curr\_n][i].first, curr\_s+1)));  }  }  }  for (int i=1; i<=n; i++) {  long long ret=2e18;  for (int j=0; j<=k ; j++) {  if(cost[i][j]==-1) continue;  ret=min(ret,cost[i][j]);  }  cout<>n>>m>>k;  for (int i=0; i>a>>b>>c;    graph[a].push\_back(make\_pair(b,c));  graph[b].push\_back(make\_pair(a,c));  }  memset(cost,-1,sizeof(cost));  djkstra();  } |

**BÀI 3. Thành phố trung tâm**

**Centre.cpp / Centre.inp / Centre.out / 1s / 1024Mb**

Theo thống kê cho biết mức độ tăng trưởng kinh tế của nước Peace trong năm 2006 rất đáng khả quan. Cả nước có tổng cộng thành phố lớn nhỏ được đánh số tuần tự từ 1 đến phát triển khá đồng đều. Giữa thành phố này là một mạng lưới gồm đường đi hai chiều, mỗi tuyến đường nối 2 trong thành phố sao cho không có 2 thành phố nào được nối bởi quá 1 tuyến đường. Trong thành phố này thì thành phố 1 và thành phố là 2 trung tâm kinh tế lớn nhất nước và hệ thống đường đảm bảo luôn có ít nhất một cách đi từ thành phố 1 đến thành phố

Tuy nhiên,cả 2 trung tâm này đều có dấu hiệu quá tải về mật độ dân số. Vì vậy, đức vua Peaceful quyết định chọn ra thêm một thành phố nữa để đầu tư thành một trung tâm kinh tế thứ ba. Thành phố này sẽ tạm ngưng mọi hoạt động thường nhật, cũng như mọi luồng lưu thông ra vào để tiến hành nâng cấp cơ sở hạ tầng. Nhưng trong thời gian sửa chữa ấy, phải bảo đảm đường đi ngắn nhất từ thành phố 1 đến thành phố không bị thay đổi, nếu không nền kinh tế quốc gia sẽ bị trì trệ.

Vị trí và đường nối giữa thành phố được mô tả như một đồ thị đỉnh cạnh. Hãy giúp nhà vua đếm số lượng thành phố có thể chọn làm trung tâm kinh tế thứ ba sao cho thành phố được chọn thỏa mãn các điều kiện ở trên

**DỮ LIỆU**

* Dòng đầu tiên ghi 2 số nguyên dương và là số thành phố và số tuyến đường.
* Dòng thứ trong số dòng tiếp theo ghi 3 số nguyên dương  với ý nghĩa tuyến đường thứ có độ dài  và nối giữa 2 thành phố

**KẾT QUẢ**

* Dòng đầu tiên ghi số tự nhiên là số lượng các thành phố có thể chọn làm trung tâm kinh tế thứ ba.
* dòng tiếp theo, mỗi dòng ghi 1 số nguyên dương là số thứ tự của thành phố được chọn ( In ra theo thứ tự tăng dần).

**VÍ DỤ**

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| 6 6  1 2 1  2 3 1  3 6 1  1 4 100  4 5 100  5 6 100 | 2  4  5 |

**THUẬT TOÁN**

* Gọi là độ dài đường đi ngắn nhất từ đỉnh 1 tới đỉnh
* Gọi là độ dài đường đi ngắn nhất từ đỉnh tới đỉnh .
* Gọi là số đường đi ngắn nhất từ đỉnh tới đỉnh
* Gọi là số đường đi ngắn nhất từ đỉnh tới đỉnh .
* Một thành phố được nhận làm trung tâm khi và chỉ khi

**CODE**

|  |
| --- |
| #include <bits/stdc++.h>  using namespace std;  const int oo = 100000001;  typedef pair <int, int > ii;  int n, m;  int ans[30001], t;  vector <ii> a[30001];  int d1[30001], d2[30001];  long long f1[30001], f2[30001];  void D1()  {  priority\_queue <ii, vector<ii>, greater<ii> > pq;  int u, i, v, du, uv;  for(i=1; i<=n; i++)  d1[i] = oo;  d1[1] = 0;  f1[1] = 1;  pq.push(ii(0,1));  while (pq.size())  {  u= pq.top().second;  du= pq.top().first;  pq.pop();  if (du!= d1[u]) continue;  for (i=0; v= a[u][i].second; i++)  {  uv= a[u][i].first;  if (d1[v] > du +uv)  {  d1[v] = du+uv;  f1[v] = f1[u];  pq.push(ii(d1[v], v));  }  else  {  if(d1[v] == du+uv)  f1[v] = f1[v] +f1[u];  }  }  }  }  void D2()  {  priority\_queue <ii, vector<ii>, greater<ii> > pq;  int u, i, v, du, uv;  for(i=1; i<=n; i++)  d2[i] = oo;  d2[n] = 0;  f2[n] = 1;  pq.push(ii(0,n));  while (pq.size())  {  u= pq.top().second;  du= pq.top().first;  pq.pop();  if (du!= d2[u]) continue;  for (i=0; v= a[u][i].second; i++)  {  uv= a[u][i].first;  if (d2[v] > du +uv)  {  d2[v] = du+uv;  f2[v] = f2[u];  pq.push(ii(d2[v], v));  }  else  {  if(d2[v] == du+uv)  f2[v] = f2[v] +f2[u];  }  }  }  }  int main()  {    scanf("%d%d",&n, &m);  for (int i=1; i<=m; i++)  {  int x, y, z;  scanf("%d%d%d", &x, &y, &z);  a[x].push\_back(ii(z, y));  a[y].push\_back(ii(z, x));  }  for (int i=1; i<=n; i++)  a[i].push\_back(ii(0,0));  D1();  D2();  for (int i=2; i<n; i++)  {  if (d1[i] + d2[i] != d1[n] || f1[i] \* f2[i] != f1[n])  {  t++;  ans[t] = i;  }  }  printf("%d\n", t);  for (int i=1; i<=t; i++)  printf("%d\n", ans[i]);    return 0;  } |

**BÀI 4. Số đường đi ngắn nhất**

**QBSCHOOL.cpp / QBSCHOOL.inp / QBSCHOOL.out / 1s / 1024Mb**

Ngày 27/11 tới là ngày tổ chức thi học kỳ I ở trường ĐH BK. Là sinh viên năm thứ nhất, Hiếu không muốn vì đi muộn mà gặp trục trặc ở phòng thi nên đã chuẩn bị khá kỹ càng. Chỉ còn lại một công việc khá gay go là Hiếu không biết đi đường nào tới trường là nhanh nhất.

Thường ngày Hiếu không quan tâm tới vấn đề này lắm cho nên bây giờ Hiếu không biết phải làm sao cả . Bản đồ thành phố là gồm có nút giao thông và con đường nối các nút giao thông này. Có 2 loại con đường là đường 1 chiều và đường 2 chiều. Độ dài của mỗi con đường là một số nguyên dương.

Nhà Hiếu ở nút giao thông 1 còn trường ĐH BK ở nút giao thông . Vì một lộ trình đường đi từ nhà Hiếu tới trường có thể gặp nhiều yếu tố khác như là gặp nhiều đèn đỏ, đi qua công trường xây dựng, ... phải giảm tốc độ cho nên Hiếu muốn biết là có tất cả bao nhiêu lộ trình ngắn nhất đi từ nhà tới trường. Bạn hãy lập trình giúp Hiếu giải quyết bài toán khó này.

**DỮ LIỆU**

* Dòng đầu tiên chứa số nguyên ;
* dòng tiếp theo mỗi dòng chứa 4 số nguyên dương trong đó:
  + nghĩa là có đường đi 1 chiều từ tới độ dài
  + nghĩa là có đường đi 2 chiều nối và độ dài .

**KẾT QUẢ**

* Gồm 2 số nguyên là độ dài đường đi ngắn nhất và số lượng đường đi ngắn nhất biết số lượng đường đi ngắn nhất không quá .

**VÍ DỤ**

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| 3 2  1 1 2 3  2 2 3 1 | 4 1 |

**THUẬT TOÁN**

* Gọi là độ dài đường đi ngắn nhất từ đỉnh 1 tới đỉnh , là số lượng đường đi ngắn nhất từ đỉnh 1 tới đỉnh
* Với mỗi cạnh từ ta cập nhật cho như sau:
  + Nếu tức phương án mới không bằng phương án cũ, ta không là gì.
  + Nếu phương án mới tốt hơn phương án cũ, thì
  + Nếu phương án mới tốt như phương án cũ, tức là có thêm lựa chọn tốt nhất, .

**CODE**

|  |
| --- |
| #include <bits/stdc++.h>  #define aaa "40"  using namespace std;  const int oo = 1e9;  typedef pair <int, int> ii; // gia tri, dinh;  int n, m;  int d[5001];  long long f[5001];  vector <ii> a[5001];  void dijkstra()  {  priority\_queue <ii, vector <ii>, greater<ii> > pq;  int u, v, du, uv, i;  pq.push(ii(0,1));  for (i=1; i<=n; i++)  d[i] = oo;  d[1] = 0;  f[1] = 1;  while (pq.size())  {  u = pq.top().second;  du = pq.top().first;  pq.pop();  if (du!= d[u]) continue;  for (i=0; v=a[u][i].second; i++)  {  uv= a[u][i].first;  if (d[v] > du + uv)  {  d[v]= du+uv;  pq.push(ii(d[v], v));  f[v] = f[u];  }  else  {  if (d[v] == du + uv)  f[v] = f[v] +f[u];  }  }  }  }  int main()  {  freopen(aaa".inp", "r", stdin);  freopen(aaa".out", "w", stdout);  scanf("%d%d", &n, &m);  int k, u, v, l;  for(int i=1; i<=m; i++)  {  scanf("%d", &k);  scanf("%d%d%d", &u, &v, &l);  a[u].push\_back(ii(l, v));  if (k==2) a[v].push\_back(ii(l,u));  }  for (int i=1; i<=n; i++)  a[i].push\_back(ii(0,0));  dijkstra();  cout << d[n] << " " << f[n] ;  return 0;  } |

**BÀI 5. Đường đi ngắn nhất và đồng dư**

**Universe.cpp / Universe.inp / Universe.out / 1s / 1024Mb**

Trong vũ trụ có vô số hành tinh, các hành tinh được đánh số thứ tự Ban đầu bạn đang đứng ở hành tinh số (trái đất). Cho dãy số nguyên trong đó tồn tại phần tử không lớn hơn Bạn có thể đi từ hành tinh tới hành tinh khi và chỉ khi tồn tại chỉ số sao cho .

Có truy vấn, mỗi truy vấn gồm 1 số nguyên , bạn cần xác định từ hành tinh ban đầu có thể đi tới hành tinh hay không?

**DỮ LIỆU**

* Dòng đầu tiên chứa hai số nguyên ;
* Dòng thứ 2 chứa số nguyên
* dòng tiếp theo mỗi dòng chứa 1 số nguyên mô tả 1 truy vấn.
  + *Dữ liệu đảm bảo tồn tại ít nhất 1 phần tử trong dãy không quá ;*
  + *Time limit 2s*

**KẾT QUẢ**

* Gồm dòng mỗi dòng ghi YES hoặc NO.

**VÍ DỤ**

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| 3 3  5 6 7  5  10  8 | YES  YES  NO |

**GIỚI HẠN**

* 30% giả thiết trong tất cả các truy vấn.
* 70% không giới hạn gì thêm.

**THUẬT TOÁN**

* Gọi .
* Xét đồ thị gồm các đỉnh Giữa 2 đỉnh tồn tại cạnh nối khi và chỉ khi tồn tại sao cho và trọng số của cạnh này là .
* Gọi là đường đi ngắn nhất từ đỉnh tới đỉnh
* Dễ dàng chứng minh được rằng số nguyên có thể biểu diễn thành tổng các số khi và chỉ khi .

**CODE**

|  |
| --- |
| ﻿#include<bits/stdc++.h>  using namespace std;  const int N = 10000;  int n, tests;  int d[N];  set<pair<int, int> > S;  set<pair<int, int> >::iterator it;  int dist[N];  int main(){  ios\_base::sync\_with\_stdio(0);  cin.tie(0);  cin >> n >> tests;  for (int i = 0; i < n; i++)  {  cin >> d[i];  }  sort(d, d + n);  for (int i = 0; i < d[0]; i++)  {  dist[i] = 2e9+1e6;  }  dist[0] = 0;  for (int i = 0; i < d[0]; i++)  {  S.insert(make\_pair(dist[i], i));  }  while (S.size())  {  it = S.begin();  int v = (\*it).second;  S.erase(it);  if (dist[v]>1e9 + 1e8)  continue;  for (int i = 0; i < n; i++)  {  int new\_val = dist[v] + d[i];  int rem = new\_val%d[0];  if (dist[rem]>new\_val)  {  S.erase(make\_pair(dist[rem], rem));  dist[rem] = new\_val;  S.insert(make\_pair(dist[rem], rem));  }  }  }  for (; tests; --tests)  {  int val;  cin >> val;  long long R = val%d[0];  if (dist[R] <= val)  {  cout << "YES\n";  }  else  {  cout << "NO\n";  }  }  return 0;  } |

**III. Bài tập luyện tập**

**BÀI 1. Cửa hàng rượu**

**Alcohol.cpp / Alcohol.inp / Alcohol.out / 1s / 1024Mb**

Thành phố có giao lộ và có con đường 2 chiều kết nối các giao lộ. Có cửa hàng rượu, mỗi cửa hàng rượu nằm tại một giao lộ nào đó. An là một thanh niên nghiện rượu, anh ta muốn biết tại mỗi giao lộ thì cửa hàng rượu gần nhất có khoảng cách bao xa?

**DỮ LIỆU**

* Dòng đầu tiên chứa số nguyên ;
* dòng tiếp theo mỗi dòng chứa 3 số nguyên mô tả con đường nối 2 giao lộ với độ dài .
* Dòng tiếp theo chứa số nguyên là số cửa hàng rượu.
* Dòng cuối cùng chứa số nguyên là chỉ số của các giao lộ có cửa hàng rượu.

**KẾT QUẢ**

* Gồm dòng, dòng thứ là khoảng cách từ giao lộ thứ tới cửa hàng rượu gần nhất.

**VÍ DỤ**

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| 3 2  1 2 4  1 3 5  1  3 | 5  9  0 |

**CODE**

Thêm đỉnh mới nối với tất cả các cửa hàng rượu với trọng số .

|  |
| --- |
| #include<bits/stdc++.h>  #define LINF 100000000000  #define rep(i,n) for(int i=0;i<n;i++)  typedef long long int ll;  using namespace std;  int n,m,u,v,x,b;  vector< pair<int,int> > graph[100009];  ll \*distances;  vector<bool> visited;  void djikstra(int start) {  distances[0] = 0;  multiset< pair<ll,int> > s;  s.insert(make\_pair(0, start));  while(!s.empty()) {  pair<ll, int> p = \*s.begin();  s.erase(s.begin());  int x = p.second;  if(visited[x]) continue;  visited[x] = true;  rep(i,graph[x].size()) {  int e = graph[x][i].first, w = graph[x][i].second;  if(distances[x] + w < distances[e]) {  distances[e] = distances[x] + w;  s.insert(make\_pair(distances[e], e));  }  }  }  }  int main() {  scanf("%d %d", &n, &m);  distances = new ll[n+1];  visited.assign(n+1, false);  fill(distances, distances+n+1, (ll)LINF);  rep(i,m) {  scanf("%d %d %d", &u,&v,&x);  graph[u].push\_back(make\_pair(v,x));  graph[v].push\_back(make\_pair(u,x));  }  scanf("%d", &b);  while(b--) {  scanf("%d", &x);  graph[0].push\_back(make\_pair(x, 0));  graph[x].push\_back(make\_pair(0, 0));  }  djikstra(0);  for(int i=1;i<=n;i++) printf("%lld\n", distances[i]);  return 0;  } |

**BÀI 2. Đường đi trên lưới ô vuông**

**Grid.cpp / Grid.inp / Grid.out / 1s / 1024Mb**

Cho một ma trận kích thước và hai số nguyên mỗi ô của ma trận có thể là ô trống được kí hiệu bởi kí tự ‘o’ hoặc có thể là ô cấm (kí hiệu bởi kí tự ‘\*’). Có truy vấn, mỗi truy vấn gồm 2 số nguyên bạn cần xác định để đi từ ô tời ô bạn cần di chuyển ít nhất bao nhiêu bước.

Mỗi bước di chuyển từ ô bạn có thể đi tới 4 ô và tất nhiên bạn không được đi ra ngoài ma trận và không được đi và ô cấm.

**DỮ LIỆU**

* Dòng đầu tiên chứa số nguyên ;
* dòng tiếp theo dòng thứ chứa 1 xâu độ dài gồm hai loại kí tự ‘o’ hoặc ‘\*’ mô tả dòng thứ của ma trận;
* Dòng tiếp theo chứa 2 số nguyên ;
* dòng tiếp theo mỗi dòng chứa 2 số nguyên mô tả 1 truy vấn.

**KẾT QUẢ**

* Gồm dòng, mỗi dòng chứa 1 số nguyên là số bước tối thiểu để đi từ ô tới ô , nếu không có đường đi ghi .

**VÍ DỤ**

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| 3 3 2  O\*O  OOO  \*OO  2 2  1 1  1 2 | 2  -1 |

**GIẢI THÍCH**

* Đường đi ngắn nhất từ ô tới ô là đo đó đáp án là 2.
* Không có đường đi từ ô tới ô .

**CODE**

|  |
| --- |
| #include<bits/stdc++.h>  #define alli(c) c.begin(),c.end()  #define forn(i,n) for(int i=0;i<n;++i)  #define pb push\_back  #define lli long long int  #define plli pair<lli,lli>  #define pii pair<int,int>  #define mp make\_pair  #define fi first  #define inf INT\_MAX  #define se second  #define mod 1000000007  using namespace std;  int n,m;  vector<string> v(1001);  vector<vector<int> > dir={{0,1},{0,-1},{1,0},{-1,0}};  vector<vector<int> > dist(1001,vector<int>(1001,10000000));  bool valid(int a,int b)  {  return (a>0 && b>0 && a<=n && b<=m);  }  void dijkastra(int x ,int y)  {  priority\_queue<pair<int,pii>,vector<pair<int,pii> >,greater<pair<int,pii> > >pq;  pq.push({0,{x,y}});  vector<vector<bool>> vis(n+1,vector<bool>(m+1,false));  dist[x][y]=0;  int a,b,d;  while(!pq.empty())  {  pair<int,pii> p=pq.top();  pq.pop();  d=p.fi,a=p.se.fi,b=p.se.se;  if(vis[a][b])  {  continue;  }  vis[a][b]=true;  forn(i,4)  {  int r=a+dir[i][0],s=b+dir[i][1];  if(valid(r,s) && v[r][s-1]=='O' && dist[a][b]+1<dist[r][s])  {  dist[r][s]=dist[a][b]+1;  pq.push({dist[r][s],{r,s}});  //cout<<r<<" "<<s<<" "<<dist[r][s]<<"\n";  }  }  }  }  int main()  {  ios\_base::sync\_with\_stdio(false);  cin.tie(NULL);  int t=1,q,x,y,a,b;  //cin>>t;  while(t--)  {  cin>>n>>m>>q;  //vector<string>v(n);  forn(i,n)  {  cin>>v[i+1];  }  cin>>x>>y;  dijkastra(x,y);  forn(i,q)  {  cin>>a>>b;  if(dist[a][b]>=10000000)  {  cout<<"-1\n";  }  else  {  cout<<dist[a][b]<<"\n";  }  }  }  return 0;  } |

**BÀI 3. Chữ cái trên lưới ô vuông**

**Pradyumn.cpp / Pradyumn.inp / Pradyumn.out / 1s /1024Mb**

Cho một ma trận , mỗi ô của ma trận là một kí tự chữ cái tiếng Anh (cả in hoa lẫn in thường). giữa 2 ô chung cạnh có một đường nối 2 chiều, độ dài của đường nối bằng giá trị tuyệt đối của hiệu 2 mã ASCII của hai kí tự ở 2 ô. Cho trước 2 ô và xác định khoảng cách của đường đi ngắn nhất từ 1 ô tới ô còn lại.

**DỮ LIỆU**

* Dòng đầu tiên chứa số nguyên là số dòng và số cột của ma trận.
* Tiếp theo là dòng mỗi dòng chứa 1 xâu kí tự chữ cái tiếng Anh.
* Dòng thứ chứa 2 số nguyên ;
* Dòng thứ chứa 2 số nguyên ;

**KẾT QUẢ**

* Gồm 1 số nguyên là khoảng cách của đường đi ngắn nhất, nếu không tồn tại đường nào thì ghi “impossible”.

**VÍ DỤ**

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| 2 2  ab  cd  2 2  1 1 | 3 |

**CODE**

|  |
| --- |
| #include "iostream"  #include "string"  #include "fstream"  #include "queue"  using namespace std;    struct point  {  int parent, child;  int weight;  point(int p, int c, int w)  {  parent = p;  child = c;  weight = w;  }  };  struct compare  {  bool operator ()(point & p1, point & p2)  {  return p1.weight > p2.weight;  }    };  int intMax=2147483647;    int dis[1500][1500];  char grid[1002][1002];  int xone , yone , xtwo, ytwo;  void shortestpath(int, int);  int n, m;  int main()  {    cin >> n >> m;  for (int i = 1; i <= n; i++)  {  char ch;  for (int j = 1; j <= m; j++)  {  dis[i][j] = intMax;  cin >> ch;  grid[i][j] = ch;  }  }  cin>> xone >> yone;  cin>> xtwo >> ytwo;  dis[xone][yone] = 0;  shortestpath(xone, yone);  cout << dis[xtwo][ytwo] << endl;    return 0;  }  void shortestpath(int r, int c)  {  priority\_queue <point, vector <point>, compare> Q;  Q.push(point(r, c, 0));  while (!Q.empty())  {  point p = Q.top();  Q.pop();    int r = p.parent;  int c = p.child;  //cout << r << " " << c<<endl;  if (r == xtwo&&c == ytwo)  break;    if (r - 1 >= 1)  {    int t = abs(int(grid[r][c]) - int(grid[r - 1][c]));  if (dis[r][c] + t < dis[r - 1][c])  {  dis[r - 1][c] = dis[r][c] + t;  Q.push(point(r - 1, c, dis[r - 1][c]));  }  }    if ( c - 1 >= 1)  {  int t = abs(int(grid[r][c]) - int(grid[r][c - 1]));  if (dis[r][c] + t < dis[r][c - 1])  {  dis[r][c - 1] = dis[r][c] + t;  Q.push(point(r, c - 1, dis[r][c - 1]));  }  }  if (c + 1 <= m)  {  int t = abs(int(grid[r][c]) - int(grid[r][c + 1]));  if (dis[r][c] + t < dis[r][c + 1])  {  dis[r][c + 1] = dis[r][c] + t;  Q.push(point(r, c + 1, dis[r][c + 1]));  }  }  if (r + 1 <= n)  {  int t = abs(int(grid[r][c]) - int(grid[r + 1][c]));  if (dis[r][c] + t < dis[r + 1][c])  {  dis[r + 1][c] = dis[r][c] + t;  Q.push(point(r + 1, c, dis[r + 1][c]));  }  }    }  } |

**BÀI 4. Sô cô la**

**Chocolate.cpp / Chocolate.inp / Chocolate.out / 1s / 1024Mb**

Thành phố gồm giao lộ (đánh số từ 1 tới ) và con đường 2 chiều kết nối các giao lộ. Có cửa hàng chocolate, mỗi cửa hàng đặt tại một giao lộ nào đó. Vị trí của An là giao lộ còn vị trí Crush là giao lộ , An biết rằng Crush thích chocolate nên sẽ không bỏ qua cơ hội này. Chocolate được bảo quản trong ngăn lạnh của cửa hàng nên có thể bảo quản trong bao lâu cũng được, tuy nhiên sau khi mua và mang đi thì chỉ bảo quản được trong đơn vị thời gian, sau đó sẽ hỏng.

Mỗi con đường có độ dài nhất định và thời gian đi qua 1 đơn vị độ dài hết 1 đơn vị thời gian. Bạn hãy giúp An xác định thời gian tối thiểu để mang chocolate cho Crush.

**DỮ LIỆU**

* Dòng đầu tiên chứa số nguyên ;
* Dòng thứ 2 chứa số nguyên là chỉ số của các giao lộ có đặt cửa hàng chocolate;
* dòng tiếp theo mỗi dòng chứa 3 số nguyên mô tả 1 con đường 2 chiều nối 2 giao lộ có độ dài ;
* Dòng cuối chứa 2 số nguyên là vị trí của Anvà Crush.
  + *Giới hạn thời gian 2s.*

**KẾT QUẢ**

* Gồm 1 số nguyên là thời gian tối thiểu đê An tặng chocolate cho Crush, nếu không thể thì ghi .

**VÍ DỤ**

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| 7 3 1 6  1  4 7 1  3 5 7  6 1 3  6 2 | -1 |

**CODE**

|  |
| --- |
| #include<bits/stdc++.h>    #define ff first  #define ss second  #define pb push\_back  #define pf push\_front  #define ins insert  #define mp make\_pair  #define sz(s) (int)s.size()  #define forp(i,a,b) for( i=a;i<=b;i++)  #define rep(i,n) for( i=0;i<n;i++)  #define ren(i,n) for( i=n-1;i>=0;i--)  #define forn(i,a,b) for( i=a;i>=b;i--)  #define w(t) while(t)  #define on cout<<"\n"  #define o2(a,b) cout<<a<<" "<<b  #define os cout<<" "  #define bitcount \_\_builtin\_popcount  #define gcd \_\_gcd  #define all(v) v.begin(),v.end()  #define mem(n,m) memset(n,m,sizeof(n))  #define pii pair<int,int>  #define pll pair<long long,long long>  #define sii set<int>  #define sll set<long long>  #define vii vector<int>  #define vll vector<long long>  #define mii map<int,int>  #define mll map<long long, long long>  #define lob lower\_bound  #define upb upper\_bound  #define ret return 0  #define present(s,x) (s.find(x) != s.end())  #define cpresent(s,x) (find(all(s),x) != s.end())  #define ford(container, it) for(\_\_typeof(container.begin()) it = container.begin(); it != container.end(); it++)  #define fors(container, it, a, b) for(\_\_typeof(container.begin()) it = a; it != b; it++)  #define boost ios\_base::sync\_with\_stdio(0);cin.tie(0);cout.tie(0);  #define MOD 1000000007  #define EPSILON 1e-9  #define PI 3.1415926535897932384626433832795028841971693993751  #define inf 999999999999999999  #define siz 100000  #define SIZ 1000000  #define SIZE 200000    typedef long long ll;  typedef long double ldo;  typedef double db ;  using namespace std;  ll pow\_(ll a,ll b,ll m)  {  a%=m;  ll res=1;  w(b)  {  if(b&1)res=(res\*a)%m;  a=(a\*a)%m;  b>>=1;  }  return res;  }  vector<pii > adj[siz+5];  int mark[siz+5];  void sp(ll dist[],int src)  {  mem(mark,0);  int i;  forp(i,1,siz)  {  dist[i]=INT\_MAX;  }  priority\_queue<pii ,vector<pii >, greater<pii > > q;  q.push(mp(0,src));  dist[src]=0;  w(!q.empty())  {  int x=q.top().ss;  q.pop();  if(mark[x]==1)  continue;  mark[x]=1;  rep(i,sz(adj[x]))  {  if(!mark[adj[x][i].ff])  {  if(dist[adj[x][i].ff]>dist[x]+adj[x][i].ss)  {  dist[adj[x][i].ff]=dist[x]+adj[x][i].ss;  q.push(mp(dist[adj[x][i].ff],adj[x][i].ff));  }  }  }  }  }  int main()  {  boost  #ifndef ONLINE\_JUDGE  freopen("input.txt.txt","r",stdin);  freopen("output.txt.txt","w",stdout);  #endif  int n,m,k,x,a,u,v,d,A,B,i;  cin>>n>>m>>k>>x;  vii vec;  rep(i,k)  {  cin>>a;  vec.pb(a);  }  w(m--)  {  cin>>u>>v>>d;  adj[u].pb(mp(v,d));  adj[v].pb(mp(u,d));  }  cin>>A>>B;  ll dist1[siz+3],dist2[siz+5];  sp(dist1,A);  sp(dist2,B);  ll ans=INT\_MAX;  rep(i,k)  {  if(dist2[vec[i]]<=x)  ans=min(ans,dist1[vec[i]]+dist2[vec[i]]);  }  if(ans==INT\_MAX)  ans=-1;  cout<<ans<<"\n";  } |

**BÀI 5. Tìm giá trị lớn nhất là nhỏ nhất trên đường đi**

**City.cpp / City.inp / City.out / 2s / 256Mb**

Một đất nước có thành phố và con đường 1 chiều. Ban đầu bạn vị trí của bạn là thủ đô . Mỗi thành phố mà bạn đi qua có bán một loại sản phẩm, thành phố thứ bán sản phẩm giá trị .

Bạn cần phải đi tới thành phố . Nếu bạn đang ở thành phố , bạn có thể chọn một con đường bất kì và đi theo con đường một chiều đó. Cứ như vậy tới khi bạn tới thành phố , hành trình lập tức kết thúc.

Trên hành trình tới thành phố , bạn chỉ mua đúng 1 sản phẩm là sản phẩm có giá trị cao nhất, gọi giá trị cao nhất này là . Rõ ràng với các hành trình khác nhau thì có thể khác nhau. Bạn cần tìm giá trị nhỏ nhất của .

**DỮ LIỆU**

* Dòng đầu tiên chứa số nguyên là số test, mỗi test có cấu trúc:
* Dòng thứ 1 chứa số nguyên ;
* Dòng thứ 2 chứa số nguyên ;
* dòng tiếp theo, mỗi dòng chứa 2 số nguyên là con đường 1 chiều từ tới .

**KẾT QUẢ**

* Gồm dòng ứng với test, mỗi test in ra số nguyên là giá trị nhỏ nhất của ứng với .

**VÍ DỤ**

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| 2  3 4 1  3 4 5  1 2  1 3  2 3  3 2  4 5 2  6 3 2 5  2 1  2 4  1 3  4 3  3 1 | 3 4 5  6 3 5 5 |

**GIẢI THÍCH**

* Trong test 2:
  + Có duy nhất đường đi tới thành phố 1, và mua sản phẩm ở thành phố 1 có giá trị 6;
  + Có duy nhất đường đi tới thành phố 4, và mua sản phẩm ở thành phố 4 có giá trị 5;
  + Có 2 đường đi tới thành phố 3 là và , với đường đi thứ nhất mua sản phẩm ở thành phố 4 có giá trị 5, với đường đi thứ hai mua sản phẩm ở thành phố 1 có giá trị 6. Do đó giá trị nhỏ nhất của là 5.

**CODE**

|  |
| --- |
| #include <bits/stdc++.h>  using namespace std;  typedef long long ll;  typedef pair<ll, int> li;  int const MAX = 1e5 + 5;  int const INF = 1e9 + 5;    int readInt()  {  int num, c;  // cin >> num;  while ((c = getchar\_unlocked()) < '-')  ;  num = c - '0';  while ((c = getchar\_unlocked()) >= '0')  {  num = (num << 3) + (num << 1) + (c - '0');  }  return num;  }    void dijkstra(int n, int start, vector<ll> val, vector<int> adj[MAX])  {  priority\_queue<int, vector<int>, greater<int>> pq;  // vector<bool> vis(n,false);  ll minDis[MAX];  for (int i = 0; i < MAX; ++i)  minDis[i] = INF;  minDis[start] = val[start];  pq.push(start);  while (!pq.empty())  {  int curr = pq.top();  pq.pop();  if (val[curr]>minDis[curr])  continue;  for (int i = 0; i < adj[curr].size(); ++i)  {  int next = adj[curr][i];    if (minDis[next] > max(minDis[curr], val[next]))  {  minDis[next] = max(minDis[curr], val[next]);  pq.push(next);  }  }  }  for (int i = 0; i < n; ++i)  cout << minDis[i] << " ";  cout << "\n";  }    int main()  {  // freopen("data.inp", "r", stdin);  // freopen("data.out", "w", stdout);    // ios\_base::sync\_with\_stdio(NULL);  // cin.tie(0);  // cout.tie(0);    int t = readInt();  while (t--)  {  vector<int> adj[MAX];  int n = readInt();  int m = readInt();  int s = readInt();  --s;  vector<ll> val;    for (int i = 0; i < n; ++i)  {  int num;  cin >> num;  val.push\_back(num);  }    for (int i = 0; i < m; ++i)  {  int u = readInt();  int v = readInt();  --u;  --v;  adj[u].push\_back(v);  }  dijkstra(n, s, val, adj);  // for (int i = 0; i < n; ++i)  // adj[i].resize(0);  }  // cout << "DONE";  return 0;  } |

**BÀI 6. Derby thành Manchester**

**Mancunian.cpp / Mancunian.inp / Mancunian.out / 2s / 256Mb**

Đó là trận Derby cuối tuần ở thành Manchester. United và City. Quỷ đỏ và Gã hàng xóm ồn ào. Các mancunian đã chờ đợi trận đấu này từ rất lâu rồi. Nhưng do mức độ nổi tiếng của trận đấu, có rất nhiều phương tiện trên các con đường và giao thông rất hỗn loạn. Để giải tỏa ách tắc, cảnh sát nghĩ ra một hệ thống các con đường 1 chiều một cách khéo léo để giao thông được thông suốt.

Thành Manchester có thể được mô hình hóa dưới dạng đồ thị gồm nút và cạnh. Các đỉnh đánh số từ 1 tới và mỗi đỉnh đại diện cho 1 giao lộ. Mỗi cạnh là đường 1 chiều nhưng hướng đi phụ thuộc vào thời điểm đi qua. Nếu có 1 con đường giữa 2 giao lộ và , hướng của con đường là vào thời điểm giây chẵn và vào thời điểm giây lẻ. Thời gian để Mancunian đi qua một con đường là đúng 1 giây. Tại mỗi giao lộ, có thể chọn 1 con đường để đi hoặc đợi một con đường nào đó đổi chiều. Trụ sở Fan Clup của Mancunian nằm ở giao lộ 1 và nơi trận đấu diễn ra là Old Trafford (Nhà hát của những giấc mơ) nằm ở giao lộ .

Có truy vấn, mỗi truy vấn gồm 1 số nguyên là thời điểm trận đấu bắt đầu, bạn cần xác định liệu các Mancunian có đến kịp giờ hay không? Thời điểm xuất phát luôn là .

**DỮ LIỆU**

* Dòng đầu tiên chứa số nguyên
* dòng tiếp theo mỗi dòng chứa 2 số nguyên mô tả 1 con đường từ tới , dữ liệu đảm bảo không có khuyên. Lưu ý rằng con đường khác với con đường .
* dòng tiếp theo mỗi dòng chứa 1 số nguyên mô tả 1 truy vấn.

**KẾT QUẢ**

* Gồm dòng ứng với truy vấn. Ghi “GGMU” nếu các Mancunian có thể đến kịp giờ, ngược lại ghi “FML”.

**VÍ DỤ**

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| 4 3 2  1 2  2 3  4 3  1  20 | FML  GGMU |

**GIẢI THÍCH**

* Trong truy vấn đầu tiên, tại thời điểm *0* , Mancunian nằm ở đỉnh *1* và hướng của các cạnh là  *1* -> *2* , *2* -> *3* và *4* -> *3* . Anh ta di chuyển đến đỉnh *2* . Đến lúc đó anh ấy đã vượt qua thời gian tối đa được phân bổ và do đó, anh ấy không đến sân vận động đúng giờ.
* Trong truy vấn thứ hai, một trong những giải pháp là như sau. Mancunian đợi ở đỉnh *1* trong 4 giây. Khi đó định hướng của các đường là *1* -> *2*, *2* -> *3* và *4* -> *3*. Anh ta di chuyển đến đỉnh *2*. Khi đó định hướng của các đường là *2* -> *1*, *3* -> *2* và *3* -> *4* . Anh ta đợi 1 giây và di chuyển đến đỉnh *3*. Khi đó, hướng của các con đường là *2* -> *1*, *3* -> *2* và *3* -> *4* . Anh ta đợi ở đỉnh *3* trong 4 giây, sau đó anh ta di chuyển đến đỉnh *4* và do đó về đích trước khi bắt đầu trận đấu, mất tổng thời gian là 12 giây.

**CODE**

|  |
| --- |
| #include<bits/stdc++.h>  using namespace std;  #define ll long long  vector<ll>o[100001];  vector<ll>e[100001];  typedef pair<ll,ll> iPair;  int main()  {  ios\_base::sync\_with\_stdio(false);  cin.tie(0);  ll n,e1,q;cin>>n>>e1>>q;  for(ll i=0;i<e1;i++)  {  ll x,y;cin>>x>>y;  e[x].push\_back(y);  o[y].push\_back(x);    }  ll dist[100001];  for(ll i=0;i<=100000;i++)  {  dist[i]=1e15;  }  ll t=0;  priority\_queue<iPair,vector<iPair>,greater<iPair> >pq;  pq.push({0,1});dist[1]=0;  while(!pq.empty())  {  ll x=pq.top().second;pq.pop();  ll o1=dist[x];  if(o1%2==0)  {  for(ll i=0;i<o[x].size();i++)  {  if(dist[o[x][i]]>dist[x]+1)  {  dist[o[x][i]]=dist[x]+1;  pq.push({dist[o[x][i]],o[x][i]});  }  }  for(ll i=0;i<e[x].size();i++)  {  if(dist[e[x][i]]>dist[x]+2)  {  dist[e[x][i]]=dist[x]+2;  pq.push({dist[e[x][i]],e[x][i]});  }  }  }  else  {  for(ll i=0;i<e[x].size();i++)  {  if(dist[e[x][i]]>dist[x]+1)  {  dist[e[x][i]]=dist[x]+1;  pq.push({dist[e[x][i]],e[x][i]});  }  }  for(ll i=0;i<o[x].size();i++)  {  if(dist[o[x][i]]>dist[x]+2)  {  dist[o[x][i]]=dist[x]+2;  pq.push({dist[o[x][i]],o[x][i]});  }  }    }    }  //////////  ll ist[100001];  for(ll i=0;i<=100000;i++)  {  ist[i]=1e15;  }  t=1;  priority\_queue<iPair,vector<iPair>,greater<iPair> >pq1;  pq1.push({1,1});ist[1]=1;  while(!pq1.empty())  {  ll x=pq1.top().second;pq1.pop();  ll o1=ist[x];  if(o1%2==0)  {  for(ll i=0;i<o[x].size();i++)  {  if(ist[o[x][i]]>ist[x]+1)  {  ist[o[x][i]]=ist[x]+1;  pq1.push({ist[o[x][i]],o[x][i]});  }  }  for(ll i=0;i<e[x].size();i++)  {  if(ist[e[x][i]]>ist[x]+2)  {  ist[e[x][i]]=ist[x]+2;  pq1.push({ist[e[x][i]],e[x][i]});  }  }  }  else  {  for(ll i=0;i<e[x].size();i++)  {  if(ist[e[x][i]]>ist[x]+1)  {  ist[e[x][i]]=ist[x]+1;  pq1.push({ist[e[x][i]],e[x][i]});  }  }  for(ll i=0;i<o[x].size();i++)  {  if(ist[o[x][i]]>ist[x]+2)  {  ist[o[x][i]]=ist[x]+2;  pq1.push({ist[o[x][i]],o[x][i]});  }  }    }    }  ll l=min(ist[n],dist[n]);//cout<<l<<"#"<<endl;  for(ll i=0;i<q;i++)  {  ll r;cin>>r;  if(r<l-1)  {  cout<<"FML";  }  else  {  cout<<"GGMU";  }  cout<<"\n";  }    } |

**BÀI 7. Mua hàng**

**Buying.cpp / Buying.inp / Buying.out / 3s / 256Mb**

Có loại đồ vật, bạn cần mua mỗi loại ít nhất một đồ vật. Có người bán hàng, mỗi người có một tập hợp các món đồ để bán, người bán hàng không bán riêng lẻ từng món đồ mình có mà chỉ bán nguyên tập hợp đồ vật với giá nhất định.

Thông tin về người bán được cho dưới dạng ma trận kích thước , trong đó nếu người bán hàng thứ có món đồ , nếu người bán hàng thứ không có món đồ thứ . Người bán hàng thứ bán các món đồ của mình với giá . Biết rằng mỗi món đồ bạn đều có thể có được từ một người bán hàng nào đó.

Hãy xác định chi phí tối thiểu bạn cần chi ra.

**DỮ LIỆU**

* Dòng đầu tiên chứa số nguyên ;
* Dòng thứ chứa số nguyên ;

**KẾT QUẢ**

* Gồm một số nguyên là chi phí tối thiểu.

**VÍ DỤ**

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| 3 3  0 1 1 2  1 0 0 3  1 1 1 9 | 5 |

**GIẢI THÍCH**

* Mua của người bán thứ nhất và thứ 2, chi phí là .

**CODE**

|  |
| --- |
| # include <bits/stdc++.h>  using namespace std;  # define fi cin  # define fo cout  # define x first  # define y second  # define ll long long  # define mp make\_pair  template < class T > T smin(T &a,T b) {if (a > b) a = b;return a;}  template < class T > T smax(T &a,T b) {if (a < b) a = b;return a;}  int main(void)  {  int n,m;  fi>>n>>m;  assert(1 <= n \* m && n \* m <= 500);  if (n <= 23)  {  static ll D[1 << 23];  const int N = 1 << n;  for (int i = 1;i < N;++i)  D[i] = 1e18;  static int M[503];  static int C[503];  for (int i = 0;i < m;++i)  {  for (int j = 0;j < n;++j)  {  int v;  fi>>v;  assert(0 <= v && v <= 1);  M[i] |= v \* (1 << j);  }  fi>>C[i];  assert(1 <= C[i] && C[i] <= 1e9);  }  for (int mask = 0;mask < N;++mask)  for (int i = 0;i < m;++i)  smin(D[M[i] | mask],D[mask] + C[i]);  if (D[N - 1] != 1e18)  fo << D[N - 1] << '\n';  else  fo << "-1\n";  }  else  {  static bitset < 505 > M[505];  static int C[505];  for (int i = 0;i < m;++i)  {  for (int j = 0;j < n;++j)  {  int v;  fi>>v;  assert(0 <= v && v <= 1);  if (v) M[i][j] = 1;  }  fi>>C[i];  assert(1 <= C[i] && C[i] <= 1e9);  }  const int N = 1 << m;  ll ans = 1e18;  for (int mask = 0;mask < N;++mask)  {  ll cost = 0;  static bitset < 505 > W;  W.reset();  for (int i = 0;i < m;++i)  if ((mask >> i) & 1)  W |= M[i],cost += C[i];  if (W.count() == n)  smin(ans,cost);  }  if (ans != 1e18)  fo << ans << '\n';  else  fo << "-1\n";  }  return 0;  } |

**BÀI 8. Hành trình tốt nhất**

**MaximumArea.cpp / MaximumArea.inp / MaximumArea.out / 2s / 256Mb**

Một ma trận kích thước , ban đầu An đứng tại ô , mỗi bước di chuyển An có thể đi sang 1 trong 4 ô chung cạnh với ô đang đứng, tất nhiên không được di chuyển ra ngoài ma trận, nếu An di chuyển vào ô thì thời gian cho bước di chuyển là .

Điểm số cho hành trình của An được định nghĩa là chỉ số dòng lớn nhất của các ô trên hành trình nhân với chỉ số cột lớn nhất trên hành trình. Hãy xác định điểm số tối đa biết thời gian cho hành trình không quá .

**DỮ LIỆU**

* Dòng đầu tiên chứa số nguyên ;
* Dòng thứ trong dòng tiếp theo chứa số nguyên .

**KẾT QUẢ**

* Gồm 1 số nguyên là điểm số tối đa của hành trình.

**VÍ DỤ**

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| 3 18  4 7 11  7 100 100  100 100 100 | 4 |

**GIẢI THÍCH**

* Hành trình của An là bắt đầu tại ô , di chuyển sang ô hết 7 đơn vị thời gian, di chuyển sang ô hết 4 đơn vị thời gian, di chuyển sang ô hết 7 đơn vị thời gian.
* Điểm số tối đa đạt được là .

**CODE**

|  |
| --- |
| #include <bits/stdc++.h>  #define fff "Maximum Area"  #define FOR(i,a,b) for(ll i=a; i<=b; ++i)  #define ROF(i,a,b) for(ll i=a; i>=b; --i)  #define pb push\_back  typedef long long int ll;  using namespace std;  void open\_file(){  freopen(fff".INP","r",stdin);  freopen(fff".OUT","w",stdout);  }  typedef pair<int, int> ii;  typedef pair<ll,ii> ip;  ll n, t, a[155][155], res;  ll d[155][155], dd[155][155];  int dx[] = {-1, 0, 0, 1};  int dy[] = {0, -1, 1, 0};    void dijkstra(){  priority\_queue<ip, vector<ip>, greater<ip> > pq;  FOR(i,1,n)  FOR(j,1,n) d[i][j] = 1e18;  d[1][1] = 0;  pq.push({ 0, {1,1} });  while(!pq.empty()){  int x = pq.top().second.first;  int y = pq.top().second.second;  ll du = pq.top().first;  pq.pop();  if(d[x][y] != du) continue;  FOR(i,0,3){  int u = x + dx[i];  int v = y + dy[i];  if(u==0||u==n+1||v==0||v==n+1) continue;  if(d[x][y] + a[u][v] < d[u][v]){  d[u][v] = d[x][y] + a[u][v];  pq.push({d[u][v],{u,v}});  }  }  }  }    void dijkstra1(int idx){  priority\_queue<ip, vector<ip>, greater<ip> > pq;  FOR(i,1,n)  FOR(j,1,n) dd[i][j] = 1e18;  FOR(i,1,n)  dd[idx][i] = d[idx][i], pq.push({dd[idx][i], {idx, i}});  while(!pq.empty()){  int x = pq.top().second.first;  int y = pq.top().second.second;  ll du = pq.top().first;  pq.pop();  if(dd[x][y] != du) continue;  FOR(i,0,3){  int u = x + dx[i];  int v = y + dy[i];  if(u==0||u==n+1||v==0||v==n+1) continue;  if(dd[x][y] + a[u][v] < dd[u][v]){  dd[u][v] = dd[x][y] + a[u][v];  pq.push({dd[u][v],{u,v}});  }  }  }  }    void dijkstra2(int idx){  priority\_queue<ip, vector<ip>, greater<ip> > pq;  FOR(i,1,n)  FOR(j,1,n) dd[i][j] = 1e18;  FOR(i,1,n)  dd[i][idx] = d[i][idx], pq.push({dd[i][idx], {i, idx}});  while(!pq.empty()){  int x = pq.top().second.first;  int y = pq.top().second.second;  ll du = pq.top().first;  pq.pop();  if(dd[x][y] != du) continue;  FOR(i,0,3){  int u = x + dx[i];  int v = y + dy[i];  if(u==0||u==n+1||v==0||v==n+1) continue;  if(dd[x][y] + a[u][v] < dd[u][v]){  dd[u][v] = dd[x][y] + a[u][v];  pq.push({dd[u][v],{u,v}});  }  }  }  }    int main(){  if(fopen(fff".inp","r")) open\_file();  ios\_base :: sync\_with\_stdio(0);  cin.tie(0); cout.tie(0);  cin >> n >> t;  FOR(i,1,n)  FOR(j,1,n) cin >> a[i][j];  dijkstra();  FOR(i,1,n){  dijkstra1(i);  FOR(x,1,n)  FOR(y,1,n)  if(dd[x][y] <= t) res = max(res, i\*y);  }  FOR(i,1,n){  dijkstra2(i);  FOR(x,1,n)  FOR(y,1,n)  if(dd[x][y] <= t) res = max(res, i\*x);  }  cout << res;  return 0;  } |

**BÀI 9. Xây dựng đồ thị mới**

**Newgraph.cpp / Newgraph.inp / Newgraph.out / 5s / 1024Mb**

Cho một cây với đỉnh, cạnh, mỗi cạnh có trọng số nhất định. Gọi tổng trọng số đường đi nối 2 đỉnh trên cây.

Ta xây dựng đồ thị một chiều như sau: Với mỗi cặp đỉnh nếu ta dựng cạnh từ với trọng số .

Xác định đường đi ngắn nhất từ đỉnh 1 tới tất cả các đỉnh trong đồ thị mới.

**DỮ LIỆU**

* Dòng đầu tiên chứa số nguyên ;
* dòng tiếp theo mỗi dòng chứa 3 số nguyên mô tả 1 cạnh của cây.

**KẾT QUẢ**

* Gồm số nguyên là đường đi ngắn nhất từ đỉnh 1 tới các đỉnh .

**VÍ DỤ**

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| 5  1 2 -2  1 3 1  2 4 5  2 5 -6 | 0 -2 -3 1 -10 |

**CODE**

|  |
| --- |
| #include <bits/stdc++.h>  using namespace std;  int n, len;  vector <pair <int, int> > adj[80010];  int in[80010], out[80010], ti, par[80010][20];  long long d[80010], dp[80010], dp1[80010], dp2[80010], a[80010];  void dfs(int u, int p) {  in[u] = ++ti;  par[u][0] = p;  for (int i = 1; i < 20; ++i)  par[u][i] = par[par[u][i - 1]][i - 1];  for (auto [v, w]: adj[u]) {  if (v == p) continue;  d[v] = d[u] + w;  dfs(v, u);  }  out[u] = ti;  }  int check(int u, int v) {  return in[u] <= in[v] && out[v] <= out[u];  }  int LCA(int u, int v) {  if (check(u, v)) return u;  for (int i = 19; i >= 0; --i)  if (par[u][i] != -1 && !check(par[u][i], v)) u = par[u][i];  return par[u][0];  }    long long dist(int u, int v) {  return d[u] + d[v] - 2 \* d[LCA(u, v)];  }  pair <long long, long long> minval[80010];  void down(int u, int p) {  dp1[u] = a[u];  minval[u] = {1e18, 1e18};  for (auto [v, w]: adj[u]) {  if (v == p) continue;  down(v, u);  dp1[u] = min(dp1[u], dp1[v] + w);  if (dp1[v] + w < minval[u].first) minval[u].second = minval[u].first, minval[u].first = dp1[v] + w;  else if (dp1[v] + w < minval[u].second) minval[u].second = dp1[v] + w;  }  }  void up(int u, int p) {  dp2[u] = min(a[u], dp2[u]);  for (auto [v, w]: adj[u]) {  if (v == p) continue;  dp2[v] = dp2[u] + w;  if (dp1[v] + w == minval[u].first) dp2[v] = min(dp2[v], minval[u].second + w);  else dp2[v] = min(dp2[v], minval[u].first + w);  up(v, u);  }  }    int main() {  // freopen("NGRAPH.inp", "r", stdin);  // freopen("NGRAPH.out", "w", stdout);    ios\_base :: sync\_with\_stdio(false);  cin.tie(0); cout.tie(0);    cin >> n;  for (int i = 1; i < n; ++i) {  int u, v, w;  cin >> u >> v >> w;  --u; --v;  adj[u].push\_back({v, w});  adj[v].push\_back({u, w});  }  len = 300;  memset(par, -1, sizeof(par));  for (int i = 1; i < n; ++i) dp[i] = 1e18;  dfs(0, -1);  for (int i = 0; i < n; ++i) {  int last = i / len \* len;  for (int j = last; j < i; ++j)  dp[i] = min(dp[i], dp[j] + dist(i, j));  if (i - last + 1 == len) {  for (int j = 0; j < n; ++j) a[j] = 1e18;  for (int j = last; j <= i; ++j) a[j] = dp[j];  down(0, -1);  up(0, -1);  for (int j = i + 1; j < n; ++j) dp[j] = min({dp[j], dp1[j], dp2[j]});  }  }  for (int i = 0; i < n; ++i) cout << dp[i] << ' ';  } |

**BÀI 10. Truyền tin trên tháp canh**

**Gondor.cpp / Gondor.inp / Gondor.out / 1s / 1024Mb**

Vùng đất huyền thoại Gondor có một hệ thống truyền tin gồm các tháp canh để báo hiệu trong trường hợp khẩn cấp.

Khi mỗi tháp canh được báo hiệu, nhân viên truyền tin ở tháp đó sẽ ngay lập tức truyền thông tin đến tất cả các tháp chưa được nhận thông tin, và theo 1 thứ tự cho trước. Việc truyền tin xảy ra đồng thời (xem ví dụ 2: ngay khi tháp 1 nhận được tin, thông tin được truyền đến cả tháp 2 và 4 thời gian tháp 2 nhận được tin là 2). Tuy nhiên nhân viên truyền tin ở tháp chỉ có thể truyền thông tin tới không quá tháp khác. Thời gian để truyền tin giữa 2 tháp bằng khoảng cách giữa 2 tháp đó. Tại thời điểm 0, thông tin bắt đầu được truyền từ tháp 1.

Tính thời gian để mỗi tháp nhận được thông tin.

**DỮ LIỆU**

* Dòng đầu tiên: là số lượng tháp. Các tháp đánh số từ 1 đến .
* dòng tiếp theo, dòng thứ gồm:
  + Số nguyên và là tọa độ của tháp trong hệ toạ độ
  + Số nguyên là số tháp mà tháp đó có thể truyền tin đi
  + số nguyên phân biệt trong khoảng 1 đến , là danh sách các tháp mà nhân viên ở tháp được yêu cầu truyền tin. Nhân viên truyền tin ở tháp này sẽ phải lần lượt truyền thông tin đi theo thứ tự trong danh sách. Không có 2 số nào trùng nhau trong danh sách, và trong danh sách thứ không chứa số
* *Dữ liệu đảm bảo không có 2 tháp nào nhận được tin tại cùng 1 thời điểm.*

**KẾT QUẢ**

* Gồm dòng, mỗi dòng chứa một số thực. Dòng thứ chứa thời gian mà tháp thứ i nhận được thông tin. Kết quả lấy chính xác 5 chữ số sau dấu phẩy.

**VÍ DỤ**

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| 4  1 1 1 2 3 4  1 2 1 4 1 3  2 1 1 2 1 4  2 2 1 3 2 1 | 0.00000  1.00000  3.00000  2.00000 |
| 5  4 3 2 5 2 4 3  4 5 1 4 1 5 3  4 4 1 1 4 5 2  2 4 1 5 2 3 1  3 4 2 2 4 3 1 | 0.00000  2.00000  4.41421  2.41421  1.41421 |

**CODE**

|  |
| --- |
| #include <bits/stdc++.h>  #define oo 10000000.0  using namespace std;  int n, x[101], y[101], s[101], t[101];    typedef pair <double, int> di;  vector <int> b[101];  vector <di> a[101];  double d[101];  priority\_queue <di, vector<di>, greater<di> > pq;    void dijkstra ()  {  double du, uv;  int u;  for (int i=1; i<=n; i++)  d[i] = oo;  d[1]=0.0;  pq.push(di(0.0, 1));    while ( pq.size())  {  u= pq.top().second;  du= pq.top().first;  pq.pop();  if (d[u] != du) continue;  for (int i=0; int v= a[u][i].second; i++)  {  if (s[u] == 0) break;  uv = a[u][i].first;  if (du < d[v])  {  s[u]--;  if (d[v] > du + uv)  {  d[v] = du +uv;  pq.push(di(d[v], v));  }  }  }  }  }  int main()  {    cin >> n;  for (int i=1; i<=n; i++)  {  cin >> x[i] >> y[i] >> s[i];  for (int j=1; j<=n-1; j++)  {  int th;  cin >> th;  b[i].push\_back(th);  }  }  for (int i=1; i<=n; i++)  for (int j=0; j<n-1; j++)  {  int u= b[i][j];  double kc = (double) sqrt((x[i] - x[u])\*(x[i] - x[u]) + (y[i] - y[u])\*(y[i] - y[u]));  a[i].push\_back(di(kc, u));  }  for (int i=1; i<=n; i++)  a[i].push\_back(di(0.0, 0));    dijkstra();  for (int i=1; i<=n; i++)  printf("%0.5f \n", d[i]);      return 0;  } |

# 

# Phần thứ ba

# KẾT LUẬN

Chuyên đề này tôi trình bày, cài đặt và giải một số bài toán áp dụng thuật toán và các tính chất thuật toán Dijkstra. Hi vọng có thể giúp ích được cho các thầy cô và học trong quá trình giảng dạy

Còn rất nhiều dạng bài tập khó mà tôi cần phải tìm hiểu hơn nữa trong chuyên đề đường đi ngắn nhất và đồ thị nói chung. Thời gian viết chuyên đề có hạn, không tránh khỏi những sai sót và hạn chế, rất mong được sự góp ý kiến của các thầy cô.

*Cảm ơn các thày cô đã đọc tài liệu!*

# TÀI LIỆU THAM KHẢO

Tài liệu sách giáo khoa chuyên tin – Hồ Sỹ Đàm chủ biên

Trang web: <http://www.vn.spoj.com>

Trang web: <https://vnoi.info/>

Trang web: https://www.hackerearth.com/practice/