1 dij+prio

1. **const** **int** inf = 0x3f3f3f3f;
2. **const** **int** maxn = 1010;
3. **int** G[maxn][maxn];
4. **int** vst[maxn];
5. **int** d[maxn];
6. **int** m, n;
8. **struct** Node{
9. **int** p, d;
10. Node(){}
11. Node(**int** pp, **int** dd):p(pp), d(dd){}
12. **friend** **bool** operator<(Node a, Node b){
13. **if**(a.p != b.p)
14. **return** a.d > b.d;//优先小的出
15. **return** a.d > b.d;//优先编号小的
16. }
17. }p;
19. **void** Dijkstra(**int** s){
20. **for**(**int** i = 1; i <= n; ++i){
21. d[i] = inf;
22. vst[i] = 0;
23. }
24. d[s] = 0;
25. priority\_queue<Node> q;
26. q.push(Node(s, d[s]));
27. **while**(!q.empty()){
28. **int** k = q.top().p;
29. q.pop();
30. **if**(vst[k])
31. **continue**;
32. vst[k] = 1;//用过的点标记不用
33. **for**(**int** j = 1; j <= n; ++j){
34. **if**(!vst[j] && G[k][j] && d[j]> d[k]+G[k][j]){//注意需要判断G[k][j]边是否存在
35. d[j] = d[k] + G[k][j];
36. q.push(Node(j, d[j]));
37. }
38. }
39. }
40. }
42. **int** main(){
43. **while**(~scanf("%d %d", &m, &n)){
44. memset(G, 0, **sizeof**(G));
45. **for**(**int** i = 0; i < m; ++i){
46. **int** x, y, v;
47. scanf("%d%d%d", &x, &y, &v);
48. **if**( G[x][y] == 0 || (G[x][y] && v < G[x][y]) ){
49. G[x][y] = v;
50. G[y][x] = v;
51. }
52. }
53. Dijkstra(1);
54. printf("%d\n", d[n]);
55. }
56. **return** 0;
57. }

1 dij+prio ed

1. **const** **int** maxn = 1e4;
2. **const** **int** inf = 0x3f3f3f3f;
4. **struct** node{
5. **int** to;
6. **int** w;
7. **int** next;
8. }e[maxn];
10. **int** head[maxn], cnt, n;
11. **int** d[maxn], vst[maxn];
13. **void** add(**int** u,**int** to,**int** w){
14. e[cnt].to = to;
15. e[cnt].w = w;
16. e[cnt].next = head[u];
17. head[u] = cnt++;
18. }
20. **struct** Node{
21. **int** u;
22. **int** dis;
23. Node(){}
24. Node(**int** uu, **int** dd){
25. u = uu;
26. dis = dd;
27. }
28. **bool** **friend** operator <(Node a,Node b){
29. **return** a.dis>b.dis;
30. }
31. };
33. **void** dijkstra(**int** s){
34. memset(d, inf, **sizeof**(d));
35. d[s]=0;
36. priority\_queue<Node> q;
37. q.push(Node(s, d[s]));
38. **while**(!q.empty()){
39. **int** k = q.top().u;//到起点距离最短还没有确定的那个点出队
40. q.pop();
41. vst[k] = 1;//当前这个节点的最短距离确定标记为1
42. **for**(**int** i = head[k]; ~i; i = e[i].next){
43. **int** to = e[i].to;
44. **int** w = e[i].w;
45. **if**(!vst[to] && d[k] + w < d[to]){
46. d[to] = d[k] + w;
47. q.push(Node(to, d[to]));
48. }
49. }
50. }
51. }
52. **int** main(){
53. memset(head, -1, **sizeof**(head));
54. **int** t, u, to, w;
55. scanf("%d%d", &t, &n);
56. **for**(**int** i = 0; i < t; ++i){
57. scanf("%d%d%d", &u, &to, &w);
58. add(u, to, w);
59. add(to, u, w);
60. }
61. dijkstra(1);
62. cout<<d[n]<<endl;
63. **return** 0;
64. }

1 dijkstra + path

1. **const** **int** inf = 100000000;
2. **const** **int** maxn = 2010;
4. **int** n,v,m;
5. **int** d[maxn];
6. **int** G[maxn][maxn];
7. **int** vst[maxn];
8. **int** path[maxn];
10. **void** Path(**int** v0, **int** v1){
11. stack<**int**> s;
12. **while**(v0 != v1){
13. s.push(v1);
14. v1 = path[v1];
15. }
16. s.push(v1);
17. **while**(!s.empty()){
18. cout<<s.top()<<"  ";
19. s.pop();
20. }
21. }
23. **void** Dijkstra(**int** s)
24. {
25. memset(vst,0,**sizeof**(vst));
26. **for**(**int** i = 1; i <= m; ++i){
27. d[i] = G[1][i];
28. vst[i] = 0;
29. }
31. d[s]=0;
32. vst[s] = 1;
33. **for**(**int** i=1;i<=m;i++){
34. **int** temp=inf;
35. **int** k=s;
36. **for**(**int** j=1;j<=m;j++){
37. **if**(!vst[j]&&d[j]<temp){
38. temp=d[j];
39. k=j;
40. }
41. }
42. vst[k]=1;
43. **for**(**int** j=1;j<=m;j++){
44. **if**(!vst[j]){
45. **if**(d[j]>d[k]+G[k][j])
46. path[j] = k;
47. d[j]=min(d[j],d[k]+G[k][j]);
48. }
49. }
50. }
51. }
53. **int** main(){
54. **while**(~scanf("%d %d", &n, &m)){
55. **for**(**int** i = 1; i <= m; ++i){
56. **for**(**int** j = 1; j <= m; ++j){
57. G[i][j] = inf;
58. }
59. }
60. **for**(**int** i = 0;i < n;i++){
61. **int** x,y,w;
62. scanf("%d %d %d", &x, &y, &w);
63. **if**( w < G[x][y] ){
64. G[x][y] = w;
65. G[y][x] = w;
66. }
67. }
68. Dijkstra(1);
70. /\*
71. cout << endl;
72. for(int i = 1; i <= 5; ++i){
73. cout<<"//////////  "<<i<<"///////"<<endl;
74. Path(1, i);
75. cout<<endl<<"now \_ d :  "<<d[i]<<endl;
76. cout<<endl<<endl;
77. }
78. cout<<endl<<endl;
79. cout << "\*\*\*\*\*\*\*\*\*\*\*" << endl;
80. \*/
81. printf("%d\n", d[m]);
82. }
83. **return** 0;
84. }

2 dijkstra

1. **const** **int** maxn = 1010;
2. **co*nst******int****inf = 0x3f3f3f3f;*
4. ***int****n;*
5. ***int****x[maxn], y[maxn];*
6. ***double****G[maxn][maxn];*
7. ***double****d[maxn];*
8. ***int****vst[maxn];*
10. ***void****Dijkstra(****int****s){*
11. ***for****(****int****i = 0; i <= n; ++i){*
12. *vst[i] = 0;*
13. *d[i] = inf;*
14. *}*
15. *vst[s] = 1;*
16. *d[s] = 0;*
17. ***for****(****int****i = 0; i < n-1; ++i){*
18. ***int****Min = inf;*
19. ***int****k = s;*
20. ***for****(****int****j = 0; j < n; ++j){*
21. ***if****(!vst[j] &&  Min > d[j]){*
22. *Min = d[j];*
23. *k = j;*
24. *}*
25. *}*
26. *vst[k] = 1;*
27. ***for****(****int****j = 0; j < n; ++j){*
28. ***if****(G[k][j])*
29. *d[j] = min(d[j],  max(d[k], G[k][j]) );*
30. *}*
31. *}*
32. *}*
34. ***int****main(){*
35. ***int****cot = 1;*
36. ***while****(~scanf("%d",&n) && n){*
37. *memset(G,0,****sizeof****(G));*
38. ***for****(****int****i = 0; i < n; ++i){*
39. *scanf("%d%d",&x[i],&y[i]);*
40. *}*
41. ***for****(****int****i = 0; i < n; ++i){*
42. ***for****(****int****j = i+1; j < n; ++j){*
43. *G[i][j] = G[j][i] = sqrt( (****double****)(x[i]-x[j]) \* (****double****)(x[i]-x[j]) +*
44. *(****double****)(y[i]-y[j]) \* (****double****)(y[i]-y[j]) );*
45. *}*
46. *}*
47. *Dijkstra(0);*
48. *printf("Scenario #%d\nFrog Distance = %.3lf\n\n",cot++,d[1]);*
49. *}*
51. ***return****0;*
52. *}*

*3 dijkstra*

1. *#define ms(x,y) memset( (x), (y), sizeof(x) )*
2. ***const******int****maxn = 1010;*
3. ***const******int****inf = 0x3f3f3f3f;*
4. ***int****t, n, m;*
5. ***int****G[maxn][maxn];*
6. ***int****vst[maxn];*
7. ***int****d[maxn];*
9. ***void****Dijkstra(****int****s){*
10. ***for****(****int****i = 0; i <= n; ++i){*
11. *d[i] = 0;*
12. *vst[i] = 0;*
13. *}*
14. *d[s] = inf;*
15. *vst[s] = 1;*
16. ***for****(****int****i = 0; i < n-1; ++i){*
17. ***int****Max = 0;*
18. ***int****k = s;*
19. ***for****(****int****j = 1; j <= n; ++j){*
20. ***if****(!vst[j] && Max < d[j]){*
21. *Max = d[j];*
22. *k = j;*
23. *}*
24. *}*
25. *vst[k] = 1;*
26. ***for****(****int****j = 1; j <= n; ++j){*
27. ***if****(G[k][j]){*
28. *d[j] = max(d[j], min(d[k], G[k][j]));*
29. *}*
30. *}*
31. *}*
32. *}*
34. ***int****main(){*
35. *scanf("%d", &t);*
36. ***for****(****int****p = 1 ; p <= t; ++p){*
37. *ms(G, 0);*
38. *scanf("%d%d",&n,&m);*
39. ***for****(****int****i = 0;i < m; ++i){*
40. ***int****x,y,v;*
41. *scanf("%d%d%d",&x,&y,&v);*
42. ***if****(G[x][y] == 0 || (G[x][y] && G[x][y] < v  ))*
43. *G[x][y] = G[y][x] = v;*
44. *}*
45. *Dijkstra(1);*
46. *printf("Scenario #%d:\n%d", p,d[n]);*
47. ***if****(p!=t)*
48. *printf("\n\n");*
49. *}*
50. ***return****0;*
51. *}*

*4 dijkstra双向 POJ 3268 Silver Cow Party*

1. *#define ms(x,y) memset((x),(y),sizeof(x))*
2. ***const******int****maxn = 1010;*
3. ***const******int****inf = 0x3f3f3f3f;*
5. ***int****n,m,x;*
6. ***int****G[maxn][maxn];*
7. ***int****ans[maxn];*
8. ***int****vst[maxn];*
9. ***int****d[maxn];*
11. ***void****Dijkstra(****int****s){*
12. ***for****(****int****i = 0; i <= n; ++i){*
13. *vst[i] = 0;*
14. *d[i] = inf;*
15. *}*
16. *vst[s] = 1;*
17. *d[s] = 0;*
18. ***for****(****int****i = 0; i < n-1; ++i){*
19. ***int****Min = inf;*
20. ***int****k = s;*
21. ***for****(****int****j = 1; j <= n; ++j){*
22. ***if****( !vst[j]&&Min > d[j]){*
23. *Min = d[j];*
24. *k = j;*
25. *}*
26. *}*
27. *vst[k] = 1;*
28. ***for****(****int****j = 1; j <= n; ++j){*
29. ***if****( G[k][j] ){*
30. ***if****( d[j] > d[k] + G[k][j] ){*
31. *d[j] = d[k] + G[k][j];*
32. *}*
33. *}*
34. *}*
35. *}*
36. *}*
38. ***void****reverse(){*
39. ***for****(****int****i = 1; i <= n; ++i){*
40. ***for****(****int****j = 1; j <= i; ++j){*
41. swap(G[i][j], G[j][i]);
42. }
43. }
44. }
46. **int** main(){
47. **while**(~scanf("%d%d%d", &n,&m,&x)){
48. ms(G, 0);
49. ms(ans, 0);
50. **int** Max = 0;
51. **for**(**int** i = 0; i < m; ++i){
52. **int** p, q, v;
53. scanf("%d%d%d", &p,&q,&v);
54. **if**(G[p][q] == 0 || (G[p][q] && G[p][q] > v)){
55. G[p][q] = v;
56. }
57. }
58. Dijkstra(x);
59. **for**(**int** i = 1; i <= n; ++i)
60. ans[i] = d[i];
61. reverse();
62. Dijkstra(x);
63. **for**(**int** i = 1; i <= n; ++i){
64. ans[i] += d[i];
65. Max = max(Max, ans[i]);
66. }
67. printf("%d\n", Max);
68. }
69. **return** 0;
70. }

5 spfa正环 POJ 1860 Currency Exchange

1. **const** **int** maxn = 210;
2. **const** **int** inf = 0x3f3f3f3f;
4. **int** n,m,s;
5. **double** v;
6. pair<**double**,**double**> G[maxn][maxn];
7. **int** vst[maxn];
8. **double** d[maxn];
9. **int** cot[maxn];
11. **int** spfa(**int** s){
12. **for**(**int** i = 0; i <= n; ++i){
13. vst[i] = 0;
14. d[i] = 0;
15. cot[i] = 0;
16. }
17. vst[s] = 1;
18. cot[s] = 1;
19. d[s] = v;
20. queue<**int**> q;
21. q.push(s);
22. **while**(!q.empty()){
23. **int** k = q.front();
24. q.pop();
25. vst[k] = 0;
26. **for**(**int** j = 1; j <= n; ++j){
27. **if**(d[j] < (d[k] - G[k][j].second) \* G[k][j].first ){
28. d[j] = (d[k] - G[k][j].second) \* G[k][j].first;
29. **if**(!vst[j]){
30. vst[j] = 1;
31. q.push(j);
32. cot[j] ++;
33. **if**(cot[j] > n)
34. **return** **true**;
35. }
36. }
37. }
38. }
39. **return** **false**;
40. }
42. **int** main(){
43. **while**(~scanf("%d%d%d%lf", &n,&m,&s,&v)){
44. memset(G, 0, **sizeof**(G));
45. **int** x, y;
46. **double** p,q;
47. **for**(**int** i = 0; i < m; ++i){
48. scanf("%d%d",&x,&y);
49. scanf("%lf%lf",&p,&q);
50. G[x][y] = make\_pair(p,q);
51. scanf("%lf%lf",&p,&q);
52. G[y][x] = make\_pair(p,q);
53. }
54. **if**(spfa(s)){
55. printf("YES\n");
56. }
57. **else**{
58. printf("NO\n");
59. }
60. }
61. **return** 0;
62. }

6 spfa负环 POJ 3259 Wormholes

1. **const** **int** maxn = 1000;
2. **const** **int** inf = 0x3f3f3f3f;
3. **int** t, n, m, w;
4. **int** G[maxn][maxn];
5. **int** d[maxn];
6. **int** cot[maxn];
7. **int** vst[maxn];
9. **bool** spfa(**int** s) {
10. queue<**int**> q;
11. **for**(**int** i = 0; i <= n; ++i) {
12. d[i] = inf;
13. vst[i] = 0;
14. cot[i] = 0;
15. }
16. d[s] = 0;
17. vst[s] = 1;
18. cot[s] = 1;
19. q.push(s);
20. **while**(!q.empty()) {
21. **int** k = q.front();
22. vst[k] = 0;
23. q.pop();
24. **for**(**int** j = 1; j <= n; ++j) {
25. **if**(G[k][j])
26. **if**(d[j] > d[k]+G[k][j]) {
27. d[j] = d[k] + G[k][j];
28. **if**(!vst[j]) {
29. vst[j] = 1;
30. q.push(j);
31. cot[j] ++;
32. **if**(cot[j] > n)
33. **return** **true**;
34. }
35. }
36. }
37. }
38. **return** **false**;
39. }
41. **int** main() {
42. scanf("%d", &t);
43. **while**(t--) {
44. memset(G, 0, **sizeof**(G));
45. scanf("%d%d%d", &n, &m, &w);
46. **int** x, y, z;
47. **for**(**int** i = 0; i < m; ++i) {
48. scanf("%d%d%d", &x, &y, &z);
49. **if**(G[x][y]){
50. G[x][y] = min(G[x][y], z);
51. G[y][x] = min(G[y][x], z);
52. }
53. **else**{
54. G[x][y] = z;
55. G[y][x] = z;
56. }
57. }
58. **for**(**int** i = 0; i < w; ++i) {
59. scanf("%d%d%d", &x, &y, &z);
60. G[x][y] = -z;
61. }
62. **if**(spfa(1))
63. printf("YES\n");
64. **else**
65. printf("NO\n");
66. }
68. **return** 0;
69. }

6 spfa负环vector+edge POJ 3259 Wormholes

1. **const** **int** maxn = 510;
2. **const** **int** inf = 0x3f3f3f3f;
4. **struct** Edge {
5. **int** v,c;
6. Edge(**int** \_v,**int** \_c):v(\_v),c(\_c) {}
7. };
9. vector<Edge> e[maxn];
10. **void** add(**int** u,**int** v,**int** c) {
11. e[u].push\_back(Edge(v,c));
12. }
14. **int** n, m, w;
15. **bool** vst[maxn];
16. **int** cot[maxn], d[maxn];
18. **bool** spfa(**int** s) {
19. memset(vst, 0, **sizeof**(vst));
20. memset(d, inf, **sizeof**(d));
21. memset(cot, 0, **sizeof**(cot));
22. vst[s] = cot[s] = 1;
23. d[s] = 0;
24. queue<**int**> q;
25. q.push(s);
26. **while**(!q.empty()) {
27. **int** u = q.front();
28. q.pop();
29. vst[u] = 0;
30. **for**(**int** i = 0 ; i < e[u].size(); ++i) {
31. **int** v = e[u][i].v;
32. **if**(d[v] > d[u] +  e[u][i].c) {
33. d[v] = d[u] +  e[u][i].c;
34. **if**(!vst[v]) {
35. vst[v] = 1;
36. q.push(v);
37. **if**(++cot[v] > n)
38. **return** **true**;
39. }
40. }
41. }
42. }
43. **return** **false**;
44. }
46. **inline** **void** init(**int** n) {
47. **for**(**int** i = 0; i <= n ; ++i) {
48. e[i].clear();
49. }
50. }
52. **int** main() {
53. **int** t;
54. scanf("%d", &t);
55. **while**(t--) {
56. scanf("%d%d%d", &n, &m, &w);
57. init(n);
58. **int** s, e, t;
59. **for**(**int** i = 0; i < m ; ++i) {
60. scanf("%d%d%d", &s, &e, &t);
61. add(s, e, t);
62. add(e, s, t);
63. }
64. **for**(**int** i = 0; i < w; ++i) {
65. scanf("%d%d%d", &s, &e, &t);
66. add(s, e, -t);
67. }
68. puts(spfa(1)? "YES": "NO");
69. }
70. **return** 0;
71. }

7 spfa POJ 1502 MPI Maelstrom

1. **const** **int** maxn = 210;
2. **const** **int** inf = 0x3f3f3f3f;
4. **int** n;
5. **int** G[maxn][maxn];
6. **int** vst[maxn];
7. **int** d[maxn];
9. **void** spfa(**int** s){
10. **for**(**int** i = 0; i <= n; ++i){
11. vst[i] = 0;
12. d[i] = inf;
13. }
14. vst[s] = 1;
15. d[s] = 0;
16. queue<**int**> q;
17. q.push(s);
18. **while**(!q.empty()){
19. **int** k = q.front();
20. q.pop();
21. vst[k] = 0;
22. **for**(**int** j = 1; j <= n; ++j){
23. **if**(d[j] > d[k] + G[k][j]){
24. d[j] = d[k] + G[k][j];
25. **if**(!vst[j]){
26. vst[j] = 1;
27. q.push(j);
28. }
29. }
30. }
31. }
32. }

35. **int** main(){
36. **while**(~scanf("%d", &n)){
37. memset(G, 0, **sizeof**(G));
38. **char** ch[12];
39. **for**(**int** i = 2; i <= n; ++i){
40. **for**(**int** j = 1; j <= i-1; ++j){
41. scanf("%s", ch);
42. **if**(ch[0] != 'x'){
43. G[i][j] = atoi(ch);
44. G[j][i] = G[i][j];
45. }
46. **else**{
47. G[i][j] = G[j][i] = inf;
48. }
49. }
50. }
52. /\*
53. for(int i = 1; i <= n; ++i){
54. for(int j = 1; j <= n; ++j){
55. printf("%4d", G[i][j]);
56. }
57. printf("\n");
58. }
59. \*/
61. spfa(1);
62. **int** Max = 0;
63. **for**(**int** i = 1; i <= n; ++i){
64. Max = max(Max, d[i]);
65. }
66. printf("%d\n", Max);
67. }
68. **return** 0;
69. }

8 floyd POJ 3660 Cow Contest

1. **const** **int** maxn = 210;
3. **int** n, m;
4. **int** G[maxn][maxn];
5. **int** ans[maxn];
7. //传递闭包
8. **void** floyd(){
9. **for**(**int** k = 1; k <= n; ++k){
10. **for**(**int** i = 1; i <= n; ++i){
11. **for**(**int** j = 1; j <= n; ++j){
12. **if**(G[i][k] && G[k][j])//检测i、j之间是否有路
13. G[i][j] = 1;
14. }
15. }
16. }
17. }
19. **int** main(){
20. **while**(~scanf("%d%d", &n, &m)){
21. memset(G, 0, **sizeof**(G));
22. memset(ans, 0, **sizeof**(ans));
23. **for**(**int** i = 0; i < m; ++i){
24. **int** x, y;
25. scanf("%d%d", &x, &y);
26. G[x][y] = 1;
27. }
29. floyd();
31. **for**(**int** i = 1; i <= n; ++i){
32. **for**(**int** j = 1; j <= n; ++j){
33. **if**(G[i][j] || G[j][i])
34. ans[i] ++;
35. }
36. }
38. **int** cot = 0;
39. **for**(**int** i = 1; i <= n; ++i){
40. **if**(ans[i] == n-1)
41. cot++;
42. }
43. printf("%d\n", cot);
44. }
45. **return** 0;
46. }

9 spfa正环 POJ 2240 Arbitrage

1. **const** **int** maxn = 160;
2. **const** **int** inf = 0x3f3f3f3f;
4. **int** n, m;
5. **int** cot;
6. map<string,**int**> mp;
7. **double** G[maxn][maxn];
8. **int** vst[maxn];
9. **double** d[maxn];
10. **int** ans[maxn];
12. **bool** spfa(**int** s){
13. **for**(**int** i = 0; i <= n; ++i){
14. vst[i] = 0;
15. d[i] = 0;
16. ans[i] = 0;
17. }
18. vst[s] = 1;
19. d[s] = 1;
20. ans[s] = 1;
21. queue<**int**> q;
22. q.push(s);
23. **while**(!q.empty()){
24. **int** k = q.front();
25. q.pop();
26. vst[k] = 0;
27. **for**(**int** j = 0; j < n; ++j){
28. **if**(d[j] < d[k]\*G[k][j]){
29. d[j] = d[k] \* G[k][j];
30. **if**(!vst[j]){
31. vst[j] = 1;
32. ans[j] ++;
33. q.push(j);
34. **if**(ans[j] > n)
35. **return** **true**;
36. }
37. }
38. }
39. }
40. **return** **false**;
41. }
43. **int** main(){
44. **int** res = 1;
45. **while**(~scanf("%d", &n) && n){
46. memset(G, 0, **sizeof**(G));
47. mp.clear();
48. string p, q;
49. **double** v;
50. cot = 0;
51. **for**(**int** i = 0; i < n; ++i){
52. cin >> p;
53. mp[p] = cot++;
54. }
55. scanf("%d", &m);
56. **for**(**int** i = 0; i < m; ++i){
57. cin >> p >> v >> q;
58. G[mp[p]][mp[q]] = v;
59. //printf("%4d%4d%6.2lf", mp[p], mp[q], v);
60. }
61. /\*
62. for(int i = 0; i < n; ++i){
63. for(int j = 0; j < n; ++j){
64. printf("%6.2lf",G[i][j]);
65. }
66. printf("\n");
67. }
68. \*/
69. **int** tag = 0;
70. **for**(**int** i = 0; i < n; ++i){
71. **if**(spfa(i))
72. tag = 1;
73. }
74. **if**(tag){
75. cout<<"Case "<<res++<<": Yes"<<endl;
76. }
77. **else**{
78. cout<<"Case "<<res++<<": No"<<endl;
79. }
80. }
81. **return** 0;
82. }

10 spfa+edge POJ 1511 Invitation Cards

1. **typedef** **long** **long** ll;
2. **const** **int** maxn = 1e6+10;
3. **const** **int** inf = 0x3f3f3f3f;
5. **struct** edge{
6. **int** to, w, next;
7. };
9. edge e[maxn];
10. **int** head[maxn];
11. **int** vst[maxn];
12. ll d[maxn];
13. **int** t, n, m;
14. **int** x[maxn], y[maxn], v[maxn];
16. **void** add(**int** i, **int** u, **int** v, **int** w){
17. e[i].to = v;
18. e[i].w = w;
19. e[i].next = head[u];
20. head[u] = i;
21. }
23. **void** spfa(**int** s){
24. **for**(**int** i = 0; i <= n; ++i){
25. vst[i] = 0;
26. d[i] = inf;
27. }
28. vst[s] = 1;
29. d[s] = 0;
30. queue<**int**> q;
31. q.push(s);
32. **while**(!q.empty()){
33. **int** k = q.front();
34. q.pop();
35. vst[k] = 0;
36. **for**(**int** j = head[k]; j != -1; j = e[j].next){
37. **if**(d[e[j].to] > d[k] + e[j].w){
38. d[e[j].to] = d[k] + e[j].w;
39. **if**(!vst[e[j].to]){
40. vst[e[j].to] = 1;
41. q.push(e[j].to);
42. }
43. }
44. }
45. }
46. }
48. **int** main(){
49. scanf("%d", &t);
50. **while**(t--){
51. scanf("%d%d", &n, &m);
52. memset(head, -1, **sizeof**(head));
53. memset(e, 0, **sizeof**(e));
54. **for**(**int** i = 0; i < m; ++i){
55. scanf("%d%d%d", &x[i], &y[i], &v[i]);
56. add(i, x[i], y[i], v[i]);
57. }
58. ll ans = 0;
59. spfa(1);
60. **for**(**int** i = 1; i <= n; ++i){
61. ans += d[i];
62. }
63. memset(head, -1, **sizeof**(head));
64. **for**(**int** i = 0; i < m; ++i){
65. add(i, y[i], x[i], v[i]);
66. }
67. spfa(1);
68. **for**(**int** i = 1; i <= n; ++i){
69. ans += d[i];
70. }
71. printf("%lld\n", ans);
72. }
73. **return** 0;
74. }

10 spfa双向 POJ 1511 Invitation Cards

1. **typedef** **long** **long** ll;
2. **const** **int** maxn = 1e6+10;
3. **const** **int** inf = 0x3f3f3f3f;
5. **struct** edge{
6. **int** to,w,next;
7. };
9. **int** head[maxn];
10. edge e[maxn];
11. **int** vst[maxn];
12. ll d[maxn];
13. **int** t,m,n;
14. **int** x[maxn], y[maxn], v[maxn];
15. ll ans[maxn];
17. **void** add(**int** top, **int** x, **int** y, **int** v){
18. e[top].next = head[x];
19. head[x] = top;
20. e[top].to = y;
21. e[top].w = v;
22. }
24. **void** spfa(**int** s){
25. **for**(**int** i = 0; i <= m; ++i){
26. d[i] = inf;
27. vst[i] = 0;
28. }
29. d[s] = 0;
30. vst[s] = 1;
31. queue<**int**> q;
32. q.push(s);
33. **while**(!q.empty()){
34. **int** k = q.front();
35. q.pop();
36. vst[k] = 1;
37. **for**(**int** j = head[k]; j != -1; j = e[j].next){
38. **if**(d[e[j].to] > d[k] + e[j].w){
39. d[e[j].to] = d[k] + e[j].w;
40. **if**(!vst[e[j].to]){
41. vst[e[j].to] = 1;
42. q.push(e[j].to);
43. }
44. }
45. }
46. }
47. }
49. **int** main(){
50. scanf("%d", &t);
51. **while**(t--){
52. memset(head, -1, **sizeof**(head));
53. memset(e, 0, **sizeof**(e));
54. memset(ans, 0, **sizeof**(ans));
55. scanf("%d%d",&n,&m);
56. **for**(**int** i = 0; i < m; ++i){
57. scanf("%d%d%d", &x[i], &y[i], &v[i]);
58. add(i,x[i],y[i],v[i]);
59. }
60. spfa(1);
61. **for**(**int** i = 1; i <= n; ++i){
62. ans[i] += d[i];
63. }
64. memset(head, -1, **sizeof**(head));
65. **for**(**int** i = 0; i < m; ++i){
66. add(i,y[i],x[i],v[i]);
67. }
68. spfa(1);
69. **for**(**int** i = 1; i <= n; ++i){
70. ans[i] += d[i];
71. }
72. ll res = 0;
73. **for**(**int** i = 1; i <= n; ++i){
74. res += ans[i];
75. }
76. printf("%lld\n", res);
77. }
78. **return** 0;
79. }

11 dij+prio ed差分约束 POJ 3159 Candies

1. **typedef** **long** **long** ll;
3. **const** **int** maxn = 1e6+10;
4. **const** **int** inf = 0x3f3f3f3f;
6. **typedef** pair<**int**,**int**> P;
7. **struct** edge{
8. **int** to, w, next;
9. };
11. edge e[maxn];
12. **int** head[maxn];
13. ll d[maxn];
14. **int** n, m;
16. **void** add(**int** i, **int** u, **int** v, **int** w){
17. e[i].to = v;
18. e[i].w = w;
19. e[i].next = head[u];
20. head[u] = i;
21. }
23. **void** spfa(**int** s){
24. **for**(**int** i = 0; i <= n; ++i)
25. d[i] = inf;
26. d[s] = 0;
27. priority\_queue<P, vector<P>, greater<P> > q;
28. q.push(P(0, s));
29. **while**(q.size()){
30. P k = q.top();
31. q.pop();
32. **int** first = k.first;
33. **int** second = k.second;
34. **if**(d[second] < first)
35. **continue**;
36. **for**(**int** j = head[second]; j != -1; j = e[j].next){
37. **if**(d[e[j].to] > d[second] + e[j].w){
38. d[e[j].to] = d[second] + e[j].w;
39. q.push(P(d[e[j].to], e[j].to));
40. }
41. }
42. }
43. }
45. **int** main(){
46. **while**(~scanf("%d%d", &n, &m)){
47. memset(head, -1, **sizeof**(head));
48. **int** x, y, v;
49. **for**(**int** i = 0; i < m; ++i){
50. scanf("%d%d%d", &x, &y, &v);
51. add(i, x, y, v);
52. }
53. ll ans = 0;
54. spfa(1);
55. printf("%lld\n", d[n]);
56. }
57. **return** 0;
58. }

12 dij+prio步行地铁 POJ 2502 Subway

1. **typedef**  **long** **long** LL;
2. **const** **int** inf = 1<<30;
3. **const** LL maxn = 410;
5. **int** sx, sy, ex, ey;
7. **struct** Point {
8. **int** x, y;
9. Point(**int** xx, **int** yy) {
10. x = xx, y = yy;
11. }
12. Point() {}
13. } e[maxn];
15. **int** k = 0;
16. **double** G[maxn][maxn];
18. **double** getDis(**double** x1, **double** y1, **double** x2, **double** y2) {
19. **return** sqrt((x1-x2)\*(x1-x2)+(y1-y2)\*(y1-y2));
20. }
22. **double** d[maxn];
23. **typedef** pair<**double**, **int**> P;
24. **bool** vst[maxn];
26. **void** Dijkstra(**int** s) {
27. priority\_queue<P, vector<P>, greater<P> > q;
28. fill(d, d+maxn, inf);
29. d[s] = 0;
30. q.push(P(d[s], s));
31. **while**(!q.empty()) {
32. P cur = q.top();
33. q.pop();
34. **int** i = cur.second;
35. **if**(vst[i]) **continue**;
36. vst[i] = **true**;
37. **for**(**int** j = 1; j <= k; ++j) {
38. **if**(d[i]+G[i][j] < d[j]) {
39. d[j] = d[i]+G[i][j];
40. q.push(P(d[j], j));
41. }
42. }
43. }
44. }
46. **int** main() {
47. **double** x, y;
48. cin >> x >> y;
49. e[++k] = Point(x, y);
50. cin >> x >> y;
51. e[++k] = Point(x, y);
52. //1为起点, 2为终点
53. **int** last= 2;
54. **while**(cin >> x >> y) {
55. //地铁建图
56. **if**(x==-1.0 && y==-1.0) {
57. **for**(**int** i = last+1; i < k; ++i) {
58. **double** dis = getDis(e[i].x, e[i].y, e[i+1].x, e[i+1].y);
59. G[i][i+1] = G[i+1][i] = dis/40000.0;
60. }
61. last = k;
62. **continue**;
63. }
64. e[++k] = Point(x, y);
65. }
66. //步行补边
67. **for**(**int** i = 1; i <= k; ++i)
68. **for**(**int** j = i+1; j <= k; ++j)
69. **if**(G[i][j] == 0)
70. G[i][j] = G[j][i] = getDis(e[i].x, e[i].y, e[j].x, e[j].y)/10000.0;
71. Dijkstra(1);
72. cout << (**int**)(d[2]\*60.0+0.5) << endl;
73. **return** 0;
74. }

13 dijkstra rank POJ 1062 昂贵的聘礼

1. **const** **int** maxn = 110;
2. **const** **int** inf = 0x3f3f3f3f;
3. **int** m, n;
4. **int** G[maxn][maxn];
5. **int** vst[maxn];
6. **int** d[maxn];
7. **int** val[maxn];
8. **int** Rank[maxn];
10. **bool** check(**int** i){
11. **if**(Rank[i] >= Rank[0] && Rank[i] - Rank[0] <= m)
12. **return** **true**;
13. **return** **false**;
14. }
16. **int** dijkstra(){
17. **for**(**int** i = 0; i < maxn; ++i){
18. vst[i] = 0;
19. d[i] = inf;
20. }
21. **for**(**int** i = 1; i <= n; ++i)
22. d[i] = val[i];
24. **for**(**int** i = 0; i < n; ++i){
25. **int** k = 0, Min = inf;
26. **for**(**int** j = 1; j <= n; ++j){
27. **if**(!vst[j] && Min > d[j]){
28. Min = d[j];
29. k = j;
30. }
31. }
33. vst[k] = 1;
34. **if**(!check(k))
35. **continue**;
36. **for**(**int** j = 1; j <= n; ++j){
37. **if**(!vst[j] && check(j) && d[j] > d[k] + G[k][j])
38. d[j] = d[k] + G[k][j];
39. }
41. }
42. **return** d[1];
43. }
45. **int** main(){
46. cin >> m >> n;
47. **for**(**int** i = 0; i < maxn; ++i)
48. **for**(**int** j = 0; j < maxn; ++j)
49. G[i][j] = inf;
51. **for**(**int** i = 1; i <= n; ++i){
52. **int** k, t, v;
53. G[i][i] = 0;
54. cin >> val[i] >> Rank[i] >> k;
55. **while**(k--){
56. cin >> t >> v;
57. G[t][i] = v;
58. }
59. }
61. **int** ans = inf;
62. **for**(**int** i = 1; i <= n; ++i){
63. Rank[0] = Rank[i];
64. ans = min(ans, dijkstra());
65. }
67. cout << ans << endl;
69. **return** 0;
70. }

14 spfa POJ 1847 Tram

1. **const** **int** maxn = 1010;
2. **const** **int** inf = 0x3f3f3f3f;
3. **int** n, a, b;
4. **int** G[maxn][maxn];
5. **int** d[maxn];
6. **int** vst[maxn];
8. **void** spfa(**int** s){
9. queue<**int**> q;
10. **for**(**int** i = 0; i <= n; ++i)
11. d[i] = inf;
12. d[s] = 0;
13. memset(vst, 0, **sizeof**(vst));
14. vst[s] = 1;
15. q.push(s);
16. **while**(!q.empty()){
17. **int** k = q.front();
18. vst[k] = 0;
19. q.pop();
20. **for**(**int** j = 1; j <= n; ++j){
21. **if**(d[j] > d[k]+G[k][j]){
22. d[j] = d[k] + G[k][j];
23. **if**(!vst[j]){
24. vst[j] = 1;
25. q.push(j);
26. }
27. }
28. }
29. }
30. }
31. **int** main(){
32. cin >> n >> a >> b;
34. **for**(**int** i = 0; i <= n; ++i)
35. **for**(**int** j = 0; j <= n; ++j){
36. **if**(i == j)
37. G[i][j] = 0;
38. **else**
39. G[i][j] = inf;
40. }
42. **for**(**int** i = 1; i <= n; ++i){
43. **int** m, v;
44. cin >> m;
45. **for**(**int** j = 0; j < m; ++j){
46. cin >> v;
47. **if**(j)
48. G[i][v] = 1;
49. **else**
50. G[i][v] = 0;
51. }
52. }
54. spfa(a);
55. **if**(d[b] >= inf)
56. cout << -1;
57. **else**
58. cout << d[b];
60. **return** 0;
61. }

15 spfa+dfs O - Extended Traffic LightOJ - 1074

1. #define inf 0x3f3f3f3f
2. **const** **int** maxn = 205;
3. **int** n, m;
4. **int** cot[maxn];
5. **int** d[maxn];
6. **int** cir[maxn];
7. **int** vst[maxn];
8. **int** head[maxn];
9. **int** val[maxn];
10. **struct** node {
11. **int** to, w;
12. **int** next;
13. **void** Node(**int** x, **int** y, **int** z) {
14. to = x;
15. w = y;
16. next = z;
17. }
18. } e[40005];
19. **void** init(**int** n, **int** m) {
20. memset(cot, 0, **sizeof**(cot));
21. memset(d, inf, **sizeof**(d));
22. memset(vst, 0, **sizeof**(vst));
23. memset(cir, 0, **sizeof**(cir));
24. memset(head, -1, **sizeof**(head));
25. }
26. **void** dfs(**int** x) {
27. cir[x] = 1;
28. **for**(**int** i = head[x]; i != -1; i = e[i].next) {
29. **if**(!cir[e[i].to]) {
30. dfs(e[i].to);
31. }
32. }
33. }
34. **void** spfa(**int** s) {
35. d[1] = 0;
36. cot[s] = 1;
37. **int** pre, nex;
38. queue<**int**> q;
39. q.push(s);
40. **while**(!q.empty()) {
41. pre = q.front();
42. q.pop();
43. vst[pre] = 0;
44. **if**(cir[pre]) **continue**;     //
45. **for**(**int** i = head[pre]; i != -1; i = e[i].next) {
46. nex = e[i].to;
47. **if**(cir[nex]) **continue**; //不加会T
48. **if**(d[nex] > d[pre] + e[i].w) {
49. d[nex] = d[pre] + e[i].w;
50. **if**(!vst[nex]) {
51. vst[nex] = 1;
52. q.push(nex);
53. **if**(++cot[nex] >= n) {
54. dfs(nex);
55. }
56. }
57. }
58. }
59. }
60. }
61. **int** mul(**int** x) {
62. **return** x\*x\*x;
63. }
64. **int** main() {
65. **int** N;
66. **int** u, v;
67. **int** Case = 1;
68. scanf("%d", &N);
69. **while**(N--) {
70. scanf("%d", &n);
71. **for**(**int** i = 1; i <= n; i++) {
72. scanf("%d", &val[i]);
73. }
74. scanf("%d", &m);
75. init(n, m);
76. **for**(**int** i = 1; i <= m; i++) {
77. scanf("%d %d", &u, &v);
78. e[i].Node(v, mul(val[v]-val[u]), head[u]);
79. head[u] = i;
80. }
81. spfa(1);
82. **int** Q, q[205];
83. scanf("%d", &Q);
84. **for**(**int** i = 0; i < Q; i++) {
85. scanf("%d", &q[i]);
86. }
87. printf("Case %d:\n", Case++);
88. **for**(**int** i = 0; i < Q; i++) {
89. **if**(d[q[i]] < 3 || d[q[i]] == inf)
90. printf("?\n");
91. **else** **if**(cir[q[i]]) printf("?\n");
92. **else** printf("%d\n", d[q[i]]);
93. }
94. }
95. **return** 0;
96. }

16 dij+prio vec 层次图 HDU 4725 The Shortest Path in Nya Graph

1. **const** **int** maxn = 500500;
2. **const** **int** inf = 0x3f3f3f3f;
4. **int** t;
5. **int** n, m, c;
6. **int** d[maxn];
7. vector<pair<**int**, **int**> > G[maxn];
9. **void** dijkstra(**int** s){
10. **for**(**int** i = 0; i <= n\*3; ++i)
11. d[i] = inf;
12. priority\_queue<pair<**int**, **int**> > q;
13. q.push(make\_pair(0, s));
14. d[1] = 0;
15. **while**(!q.empty()){
16. **int** u = q.top().second;
17. q.pop();
18. **for**(**int** i = 0; i < G[u].size(); ++i){
19. **int** v = G[u][i].first;
20. **int** w = G[u][i].second;
21. **if**(d[v] > d[u] + w){
22. d[v] = d[u] + w;
23. q.push(make\_pair(-d[v], v));
24. }
25. }
26. }
27. }
28. // 假设1-n为实点，n+i\*2-1为i点所在层的抽象入点,n+i\*2为i点所在层的抽象出点。
29. // 抽象入点到层内各点距离为0，抽象出点到层内各点距离为0。
30. // 抽象出点到下一层抽象入点距离为c。
31. **int** main(){
32. scanf("%d", &t);
33. **for**(**int** T = 1; T <= t; ++T){
34. **for**(**int** i = 0; i <= n\*3; ++i)
35. G[i].clear();
36. scanf("%d%d%d", &n, &m, &c);
37. **for**(**int** i = 1; i <= n; ++i){
38. **int** v;
39. scanf("%d", &v);
40. G[n + v\*2 - 1].push\_back(make\_pair(i, 0));
41. G[i].push\_back(make\_pair(n + v\*2, 0));
42. }
43. **for**(**int** i = 0; i < m; ++i){
44. **int** u, v, w;
45. scanf("%d%d%d", &u, &v, &w);
46. G[u].push\_back(make\_pair(v, w));
47. G[v].push\_back(make\_pair(u, w));
48. }
49. **for**(**int** i = 1; i <= n; ++i){
50. **int** u = n + i\*2;
51. **if**(i > 1)
52. G[u].push\_back(make\_pair(u-3, c));
53. **if**(i < n)
54. G[u].push\_back(make\_pair(u+1, c));
55. }
56. dijkstra(1);
57. printf("Case #%d: %d\n", T, d[n] >= inf ? -1 : d[n]);
58. }
59. **return** 0;
60. }

18spfa建图转化 HDU 4370 0 or 1

1. **const** **int** maxn = 310;
2. **const** **int** inf = 0x3f3f3f3f;
4. **int** n;
5. **int** vst[maxn];
6. **int** d[maxn];
7. **int** G[maxn][maxn];
9. **void** spfa(**int** s){
10. queue<**int**> q;
11. memset(vst, 0, **sizeof**(vst));
12. **for**(**int** i = 1; i <= n; ++i){
13. **if**(i == s)
14. d[i] = inf;
15. **else**{
16. d[i] = G[s][i];
17. q.push(i);
18. vst[i] = 1;
19. }
20. }
21. **while**(!q.empty()){
22. **int** k = q.front();
23. q.pop();
24. vst[k] = 0;
25. **for**(**int** j = 1; j <= n; ++j){
26. **if**(d[j] > d[k] + G[k][j]){
27. d[j] = d[k] + G[k][j];
28. **if**(!vst[j]){
29. q.push(j);
30. vst[j] = 1;
31. }
32. }
33. }
34. }
35. }
37. **int** main(){
38. **while**(~scanf("%d", &n)){
39. **for**(**int** i = 1; i <= n; ++i)
40. **for**(**int** j = 1; j <= n; ++j)
41. scanf("%d", &G[i][j]);
42. spfa(1);
43. **int** v1, v2, v3;
44. v1 = d[n];
45. v2 = d[1];
46. spfa(n);
47. v3 = d[n];
48. **int** ans = min(v1, v2+v3);
49. printf("%d\n", ans);
50. }
51. **return** 0;
52. }

19 spfa+ed 差分约束 POJ 3169 Layout

1. **typedef** **long** **long** ll;
2. **const** **int** maxn = 1e6+10;
3. **const** **int** inf = 0x3f3f3f3f;
5. **struct** edge{
6. **int** to, w, next;
7. };
9. edge e[maxn];
10. **int** head[maxn];
11. **int** vst[maxn];
12. **int** cot[maxn];
13. ll d[maxn];
14. **int** t, n, m;
16. **void** add(**int** i, **int** u, **int** v, **int** w){
17. e[i].to = v;
18. e[i].w = w;
19. e[i].next = head[u];
20. head[u] = i;
21. }
22. **bool** spfa(**int** s){
23. **for**(**int** i = 0; i <= n; ++i){
24. vst[i] = 0;
25. d[i] = inf;
26. cot[i] = 0;
27. }
28. vst[s] = 1;
29. d[s] = 0;
30. cot[s] = 1;
31. queue<**int**> q;
32. q.push(s);
33. **while**(!q.empty()){
34. **int** k = q.front();
35. q.pop();
36. vst[k] = 0;
37. **for**(**int** j = head[k]; j != -1; j = e[j].next){
38. **if**(d[e[j].to] > d[k] + e[j].w){
39. d[e[j].to] = d[k] + e[j].w;
40. **if**(!vst[e[j].to]){
41. vst[e[j].to] = 1;
42. q.push(e[j].to);
43. cot[e[j].to] ++;
44. **if**(cot[e[j].to] > n)
45. **return** **false**;
46. }
47. }
48. }
49. }
50. **return** **true**;
51. }
52. **int** main(){
53. **while**(~scanf("%d%d%d", &n, &m, &t)){
54. memset(head, -1, **sizeof**(head));
55. memset(e, 0, **sizeof**(e));
56. **int** x, y, z;
57. **for**(**int** i = 0; i < m; ++i){
58. scanf("%d%d%d", &x, &y, &z);
59. add(i, x, y, z);
60. }
61. **for**(**int** i = m; i < m+t; ++i){
62. scanf("%d%d%d", &x, &y, &z);
63. add(i, y, x, -z);
64. }
65. **if**(spfa(1))
66. printf("%d\n", d[n] == inf ? -2 : d[n]);
67. **else**
68. printf("-1\n");
69. }
70. **return** 0;
71. }