spfa/

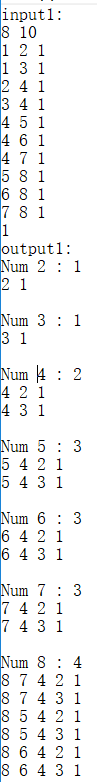
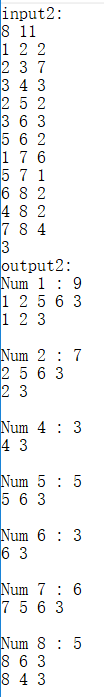
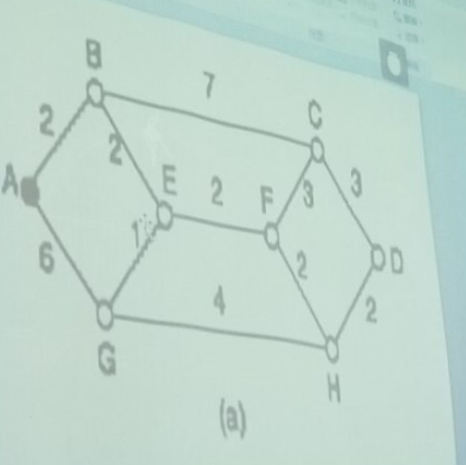
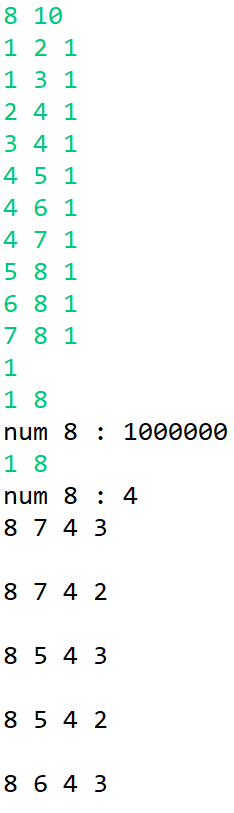
dijkstra

static

spfa/

dijkstra

static

java

dynamic

Dijkstra：

1. //dijkstra+堆优化(提高速度):O(nlogn)
2. #include <cstdio>
3. #include <cstdlib>
4. #include <queue>
5. #include <algorithm>
6. #include <iostream>
7. **using** **namespace** std;
8. **const** **long** maxn=1e4;
9. **const** **long** inf=1e9;
11. **struct** node
12. {
13. **long** d,len;
14. **struct** node \*next;
15. }\*e[maxn];
16. **struct** rec
17. {
18. **long** d;
19. **struct** rec \*next;
20. }\*nex[maxn];
22. **long** n,s,dist[maxn],index,pr[maxn];
23. **bool** vis[maxn];
25. **struct** cmp
26. {
27. **bool** operator ()(**long** i,**long** j)
28. {
29. **return** dist[i]>dist[j];
30. }
31. };
32. priority\_queue<**long**,vector<**long**>,cmp> st;
34. **void** dijkstra(**long** d)
35. {
36. **struct** node \*p;
37. **struct** rec \*q;
38. **long** i,j;
39. **for** (i=1;i<=n;i++)
40. {
41. dist[i]=inf;
42. vis[i]=0;
43. }
44. dist[d]=0;
45. st.push(d);
46. **for** (j=1;j<n;j++)
47. {
48. **while** (vis[st.top()])
49. st.pop();
50. d=st.top(); //为源路由器要依次访问的路由器
51. vis[d]=1;
52. p=e[d];
53. **while** (p)
54. {
55. **if** (!vis[p->d])
56. {
57. **if** (dist[p->d]>dist[d]+p->len)
58. {
59. dist[p->d]=dist[d]+p->len;
60. st.push(p->d);
61. nex[p->d]=(**struct** rec \*) malloc (**sizeof**(**struct** rec));
62. nex[p->d]->d=d;
63. nex[p->d]->next=NULL;
64. }
65. **else** **if** (dist[p->d]==dist[d]+p->len)
66. {
67. q=(**struct** rec \*) malloc (**sizeof**(**struct** rec));
68. q->d=d;
69. q->next=nex[p->d];
70. nex[p->d]=q;
71. }
72. }
73. p=p->next;
74. }
75. }
76. }
78. **void** print(**long** d,**long** index)
79. {
80. **struct** rec \*q;
81. pr[index]=d;
82. **if** (d!=s)
83. {
84. q=nex[d];
85. **while** (q)
86. {
87. print(q->d,index+1);
88. q=q->next;
89. }
90. }
91. **else**
92. {
93. **for** (**long** i=1;i<=index;i++)
94. printf("%ld ",pr[i]);
95. printf("\n");
96. }
97. }
99. **int** main()
100. {
101. **struct** node \*p;
102. **long** i,m,x,y,z;
103. //n points & m edges
104. //points: 1~n
105. scanf("%ld%ld",&n,&m);
106. **for** (i=1;i<=n;i++)
107. e[i]=NULL;
108. **for** (i=1;i<=m;i++)
109. {
110. //edge:two-way
111. scanf("%ld%ld%ld",&x,&y,&z);
112. p=(**struct** node \*) malloc (**sizeof**(**struct** node));
113. p->d=y;
114. p->len=z;
115. p->next=e[x];
116. e[x]=p;
118. p=(**struct** node \*) malloc (**sizeof**(**struct** node));
119. p->d=x;
120. p->len=z;
121. p->next=e[y];
122. e[y]=p;
123. }
124. scanf("%ld",&s);    //source index
125. dijkstra(s);
126. **for** (i=1;i<=n;i++)
127. **if** (i!=s)
128. {
129. printf("\nNum %ld : %ld\n",i,dist[i]);
130. index=0;
131. print(i,1);
132. }
133. **return** 0;
134. }

Spfa：(其它部分与Dijkstra相同 previous Line125. spfa(s) )

1. **long** queue[10000];
2. //queue:最大长度为n 为源路由器要依次访问的路由器
4. **void** spfa(**long** d)
5. {
6. **struct** node \*p;
7. **struct** rec \*q;
8. **long** i,head,tail;
9. **for** (i=1;i<=n;i++)
10. {
11. dist[i]=inf;
12. vis[i]=0;
13. }
14. dist[d]=0;
15. head=0; tail=1;
16. queue[1]=d; vis[1]=1;
17. **while** (head!=tail)
18. {
19. head=(head+1)%n;
20. d=queue[head];
21. p=e[d];
22. **while** (p)
23. {
24. **if** (dist[p->d]>dist[d]+p->len)
25. {
26. **if** (!vis[p->d])
27. {
28. tail=(tail+1)%n;
29. queue[tail]=p->d;
30. }
31. dist[p->d]=dist[d]+p->len;
32. nex[p->d]=(**struct** rec \*) malloc (**sizeof**(**struct** rec));
33. nex[p->d]->d=d;
34. nex[p->d]->next=NULL;
35. }
36. **else** **if** (dist[p->d]==dist[d]+p->len)
37. {
38. q=(**struct** rec \*) malloc (**sizeof**(**struct** rec));
39. q->d=d;
40. q->next=nex[p->d];
41. nex[p->d]=q;
42. }
43. p=p->next;
44. }
45. vis[d]=0;
46. }
47. }

java：

1. **import** java.util.Scanner;
3. **class** Route **extends** Thread {
4. **int**[] point=**new** **int**[100];
5. **int**[] len=**new** **int**[100];
6. **int**[] next=**new** **int**[100];
7. **int** num,dist=100000;
8. **static** **int**[] pr=**new** **int**[100];
10. **public** Route(**int** \_num,**int** \_dist,**int**[] \_point,**int**[] \_len) {
11. num=\_num;
12. point=\_point;
13. len=\_len;
14. next[0]=0;
15. dist=\_dist;
16. }
17. **public** **void** run() {
18. //延迟，使所有路由器对象创建完毕
19. **try** {
20. Thread.sleep(1000);
21. } **catch**(Exception e) {
22. e.printStackTrace();
23. }
24. **int** i,j,l;  //i : here, not global
25. **while** (**true**) {
26. **for** (i=1;i<=point[0];i++) {
27. l=Route\_shortpath.route[point[i]].getdist();
28. **if** (dist>l+len[i]) {
29. dist=l+len[i];
30. next[0]=1;
31. next[1]=point[i];
32. }
33. **else** **if** (dist==l+len[i]) {
34. **for** (j=1;j<=next[0];j++)
35. **if** (next[j]==point[i])
36. **break**;
37. **if** (j==next[0]+1) {
38. next[0]++;
39. next[next[0]]=point[i];
40. }
41. }
42. }
43. **try** {
44. Thread.sleep(100);  //每隔100ms执行一次
45. } **catch**(Exception e) {
46. e.printStackTrace();
47. }
48. }
49. }
51. **public** **int** getdist() {
52. **return** dist;
53. }
54. **public** **void** getnext(**int** index) {
55. **int** i;
56. **if** (num==Route\_shortpath.s) {
57. **for** (i=1;i<index;i++)
58. System.out.print(pr[i]+" ");
59. System.out.println("\n");
60. }
61. **else** {
62. **for** (i=1;i<=next[0];i++) {
63. pr[index]=num;
64. Route\_shortpath.route[next[i]].getnext(index+1);
65. }
66. }
67. }
68. **public** **void** getresult() {
69. System.out.println("num "+num+" : "+dist);
70. getnext(1);
71. }
72. }
74. **public** **class** Route\_shortpath {
75. **static** **int** s;
76. **static** Route[] route=**new** Route[100];
77. **public** **static** **void** main(String[] args1) **throws** InterruptedException {
78. **int** n,m,x,y,z,i,mode;
79. **int**[][] point=**new** **int**[100][100];
80. **int**[][] len=**new** **int**[100][100];
82. Scanner keyin=**new** Scanner(System.in);
83. n=keyin.nextInt();
84. m=keyin.nextInt();
85. **for** (i=1;i<=n;i++)
86. point[i][0]=0;
87. **for** (i=1;i<=m;i++) {
88. x=keyin.nextInt();
89. y=keyin.nextInt();
90. z=keyin.nextInt();
91. point[x][0]++;
92. point[x][point[x][0]]=y;
93. len[x][point[x][0]]=z;
94. point[y][0]++;
95. point[y][point[y][0]]=x;
96. len[y][point[y][0]]=z;
97. }
98. s=keyin.nextInt();
99. **for** (i=1;i<=n;i++) {
100. **if** (i==s)
101. route[i]=**new** Route(i,0,point[i],len[i]);
102. **else**
103. route[i]=**new** Route(i,1000000,point[i],len[i]);
104. route[i].start();
105. }
106. Thread.sleep(1000);
108. **while** (**true**) {
109. mode=keyin.nextInt();
110. **if** (mode==0)
111. **break**;
112. x=keyin.nextInt();
113. route[x].getresult();
114. //之后加上修改功能
115. }
116. keyin.close();
117. }
118. }
119. /\*
120. 2 1
121. 1 2 3
122. 1
123. 1 2
124. \*/