

Term Project

Algorithm and Practices

19011824 이연희

Question 1

My code

I used list structures

num is "the number of a village or castle"

print is "how long it took from the starting point"

pre is "front num"

pre_num is "taking time from the preceding num"

Code for adding path

```
Project81 (전역 범위)
1  #pragma warning(disable:4996)
2  #include <stdio.h>
3  #include <stdlib.h>
4  int N;
5  int A, B, M, L, K;
6  typedef struct vertex {
7      int num;
8      int flag;
9      int print;
10     int pre;
11     int pre_num;
12     struct vertex *next;
13 }v;
14 typedef struct edge {
15     int a, b;
16     int weight;
17     int pass;
18     struct edge *enext;
19     v *v1;
20     v *v2;
21 }e;
22
23 void getvnode(v **p) {
24     (*p) = (v *)malloc(sizeof(v));
25     (*p)->next = NULL;
26     (*p)->flag = 0;
27     (*p)->print = 0;
28 }
29 void getenode(e **p) {
30     (*p) = (e *)malloc(sizeof(e));
31     (*p)->v1 = NULL;
32     (*p)->v2 = NULL;
33     (*p)->enext = NULL;
34     (*p)->pass = 0;
35 }
```

```
36 void addenode(int a, int b, int w, e *H_edge) {
37     e *q, *p;
38     getenode(&q);
39     q->a = a;
40     q->b = b;
41     q->weight = w;
42     for (p = H_edge; p->enext != NULL; p = p->enext);
43     p->enext = q;
44 }
```

Question 1

My code

```
43 void graph(v **V, e *H_edge, int *o) {
44     e *p, *q = NULL;
45     int i = N, j, flag = 0, k, next_num, F, min, w;
46     V[N - 1]->print = 0;
47     V[N - 1]->flag = 1;
48     while (1) {
49         flag = 0; F = 0; min = 1000;
50         for (p = H_edge->enext; p != NULL; p = p->enext) {
51             if (p->pass == 1)
52                 continue;
53             if (p->a == i || p->b == i) {
54                 w = 0;
55                 if (p->a <= A+1 && p->b <= A+1)
56                     w = p->weight;
57                 if (p->a == i) {
58                     j = p->b;
59                     k = (V[i - 1]->print) + (p->weight);
60                 }
61             }
62             else {
63                 j = p->a;
64                 k = (V[i - 1]->print) + (p->weight);
65             }
66             if (V[j - 1]->flag == 1) {
67                 continue;
68             }
69             if ((V[j - 1]->print == 0) || (V[j - 1]->print > k)) {
70                 V[j - 1]->print = k;
71                 V[j - 1]->pre = i;
72                 o[j - 1] = w;
73                 p->pass = 1;
74                 if (min > k) {
75                     min = k;
76                     flag = 1;
77                     next_num = j;
78                     q = p;
79                 }
80             }
81         }
82     }
83 }
```

Code for finding minimum path

```
85     if (flag == 1) {
86         for (F = 0; F < N; F++) {
87             if (V[F]->flag == 0) {
88                 if (min > V[F]->print && V[F]->print != 0) {
89                     min = V[F]->print;
90                     next_num = F + 1;
91                     flag = 1;
92                 }
93             }
94         }
95         V[next_num - 1]->flag = 1;
96         i = next_num;
97     }
98     else if (flag == 0) {
99         min = 1000;
100         for (F = 0; F < N; F++) {
101             if (V[F]->flag == 0) {
102                 if (min > V[F]->print && V[F]->print != 0) {
103                     min = V[F]->print;
104                     i = F + 1; j = V[F]->pre;
105                     flag = 1;
106                 }
107             }
108         }
109         if (flag == 0)
110             break;
111         else {
112             V[i - 1]->flag = 1;
113         }
114     }
115 }
116 }
```

Question 1

My code

```
117 void setting() {
118     int i, sum;
119     int n, m, l;
120     v **V;
121     e *H_edge;
122     e *p;
123     int k;
124     scanf("%d %d %d %d %d", &A, &B, &M, &L, &K);
125     k = L;
126     N = A + B;
127     V = (v **)malloc(sizeof(v *)*N);
128     getenode(&H_edge);
129     for (i = 0; i < N; i++) {
130         getvnnode(&V[i]);
131         V[i]->num = i + 1;
132     }
133     for (i = 0; i < M; i++) {
134         scanf("%d %d %d", &n, &m, &l);
135         addenode(n, m, l, H_edge);
136     }
137     int *o;
138     o = (int *)malloc(sizeof(int)*M);
139     for (i = 0; i < M; i++) {
140         o[i] = 0;
141     }
142     graph(V, H_edge, o);
143     sum = 0;
144     int max = 0, flag = 0, st = 0;
```

```
145     while (1) {
146         i = st;
147         sum = 0;
148         k = L;
149         for (; i < M; i++) {
150             if (o[i] == 0) {
151                 flag = 1;
152                 break;
153             }
154             if (o[i] <= k) {
155                 sum += o[i];
156                 k -= o[i];
157             }
158             else {
159                 break;
160             }
161         }
162         if (max < sum)
163             max = sum;
164         if (flag == 1)
165             break;
166         st++;
167     }
168     int output = (V[0]->print) - max;
169     printf("%d\n", output);
170 }
```

Main code

```
171 void main() {
172     int T, i;
173     scanf("%d", &T);
174     for (i = 0; i < T; i++) {
175         setting();
176     }
177 }
```

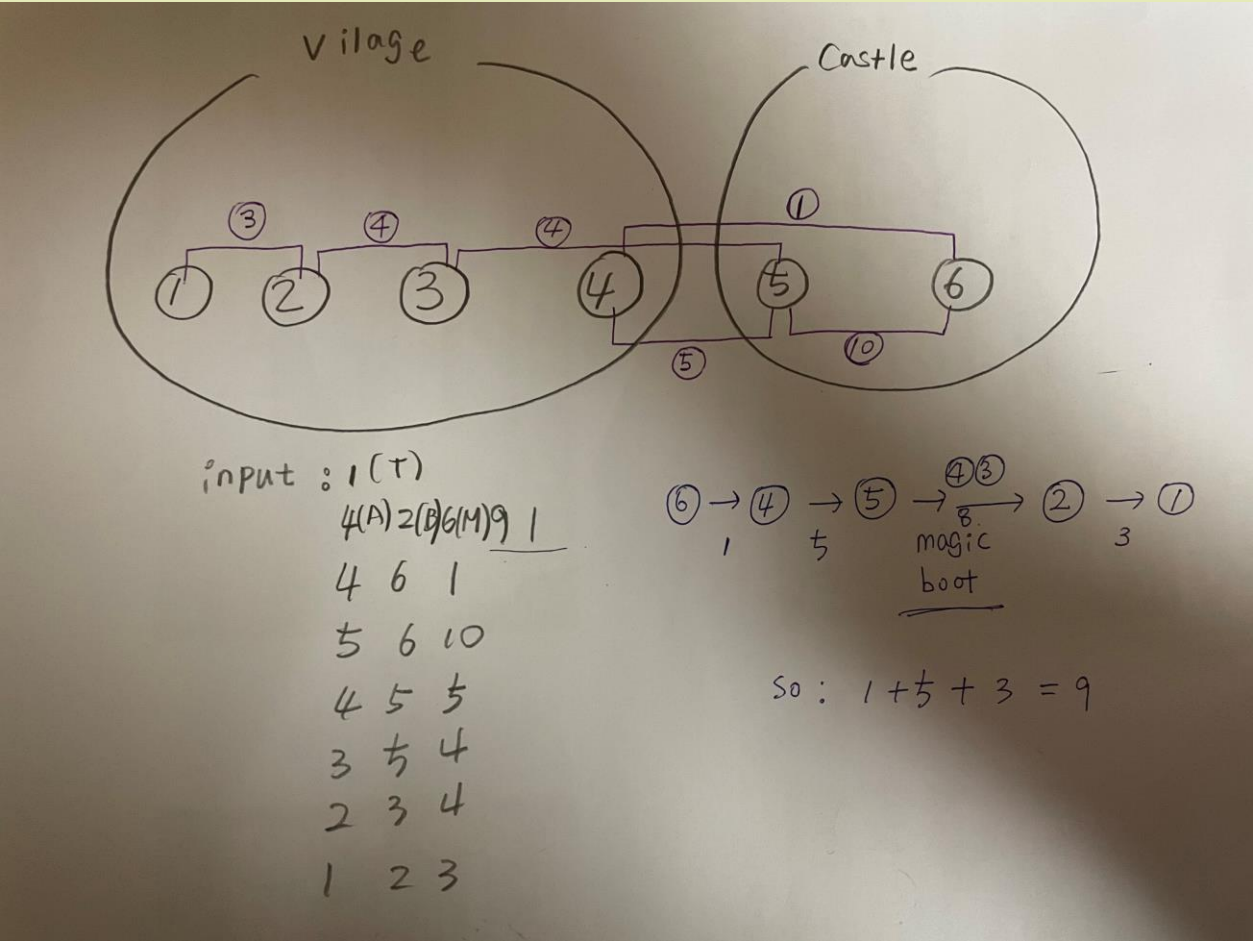
Code to enter A,B,M,L,K and call a graph() function, and apply magic boots

Question 1

Result

Microsoft Visual Studio 디버그 콘솔

```
1
4 2 6 9 1
4 6 1
5 6 10
4 5 5
3 5 4
2 3 4
1 2 3
9
C:\Users\yeonhee\source\repos\Project81
이 창을 닫으려면 아무 키나 누르세요.
```



Question 2

My code

I used list structures

"min" has the smallest number of maximum capacity to reach the current peak.

```
1  #pragma warning(disable:4996)
2  #include <stdio.h>
3  #include <stdlib.h>
4  int N, R, S, D, T;
5  typedef struct vertex {
6      int num;
7      int min;
8      int flag;
9      int print;
10     struct vertex *next;
11 }v;
12 typedef struct edge {
13     int a, b;
14     int weight;
15     int pass;
16     struct edge *enext;
17 }e;
18 v **V; e *H_edge;
19 void getvnode(v **p) {
20     (*p) = (v *)malloc(sizeof(v));
21     (*p)->next = NULL;
22     (*p)->flag = 0;
23     (*p)->print = 0;
24 }
25 void getenode(e **p) {
26     (*p) = (e *)malloc(sizeof(e));
27     (*p)->enext = NULL;
28     (*p)->pass = 0;
29 }
```

Code for adding path

```
30 void addenode(int a, int b, int w) {
31     e *q, *p;
32     getenode(&q);
33     q->a = a;
34     q->b = b;
35     q->weight = w;
36     for (p = H_edge; p->enext != NULL; p = p->enext);
37     p->enext = q;
38 }
```

Question 2

My code

```
39 void graph() {
40     e *p, *q = NULL;
41     int i = S, j, flag = 0, k, next_num, F, min=1000, max;
42     V[S - 1]->print = 0;
43     V[S - 1]->flag = 1;
44     V[D - 1]->min = 0;
45     while (1) {
46         flag = 0; F = 0;
47         max = 0;
48         min = 1000;
49         int flagg = 0;
50         for (p = H_edge->enext; p != NULL; p = p->enext) {
51             if (p->pass == 1)
52                 continue;
53             if (p->a == i || p->b == i) {
54                 if (p->a == i)
55                     j = p->b;
56                 else
57                     j = p->a;
58                 if (max < p->weight) {
59                     p->pass = 1;
60                     V[j-1]->print = p->weight;
61                     flag = 1;
62                     max = p->weight;
63                     F = j;
64                     if (min > p->weight)
65                         min = p->weight;
66                     if (j == D) {
67                         if (V[j-1]->min < min)
68                             V[j - 1]->min = min;
69                         F = S;
70                     }
71                 }
72             }
73         }
74         if (flag == 0)
75             break;
76         i = F;
77     }
78 }
```

Code for finding the best route

Choose the route that can carry the most people from the vertex of S to the vertex of D.

Initialization code for receiving the following inputs

```
79 void ini() {
80     int i;
81     e *p;
82     for (i = 0; i < N; i++) {
83         V[i]->print = 0;
84         V[i]->flag = 0;
85     }
86     for (p = H_edge->enext; p != NULL; p = p->enext) {
87         p->pass = 0;
88     }
89 }
```

Question 2

My code

```
90 int main() {
91     int i, n, m, l, j, num;
92     scanf("%d %d", &N, &R);
93     V = (v **)malloc(sizeof(v *)*N);
94     getenode(&H_edge);
95     for (i = 0; i < N; i++) {
96         getvnnode(&V[i]);
97         V[i]->num = i + 1;
98     }
99     for (i = 0; i < R; i++) {
100         scanf("%d %d %d", &n, &m, &l);
101         addenode(n, m, l);
102     }
103     printf("\n");
104     for (i = 1; i++) {
105         scanf("%d %d", &S, &D);
106         if (S == 0 && D == 0)
107             break;
108         scanf("%d", &T);
109         graph();
110         num = T / (V[D - 1]->min);
111         if (T%(V[D - 1]->min) != 0)
112             num++;
113         printf("Scenario #%d\n", i);
114         printf("Minimum Number of Trips = %d\n\n", num);
115         ini();
116     }
117     return 0;
118 }
119 }
```

Main code

V is the vertex and N dynamic allocation.

Add path by calling up addnode() function.

Repeat until 0 is entered in S and D.

T is the number of people, and calls the graph() function to find the best path.

And get how many times guide go back and forth to num.

Question 2

Result

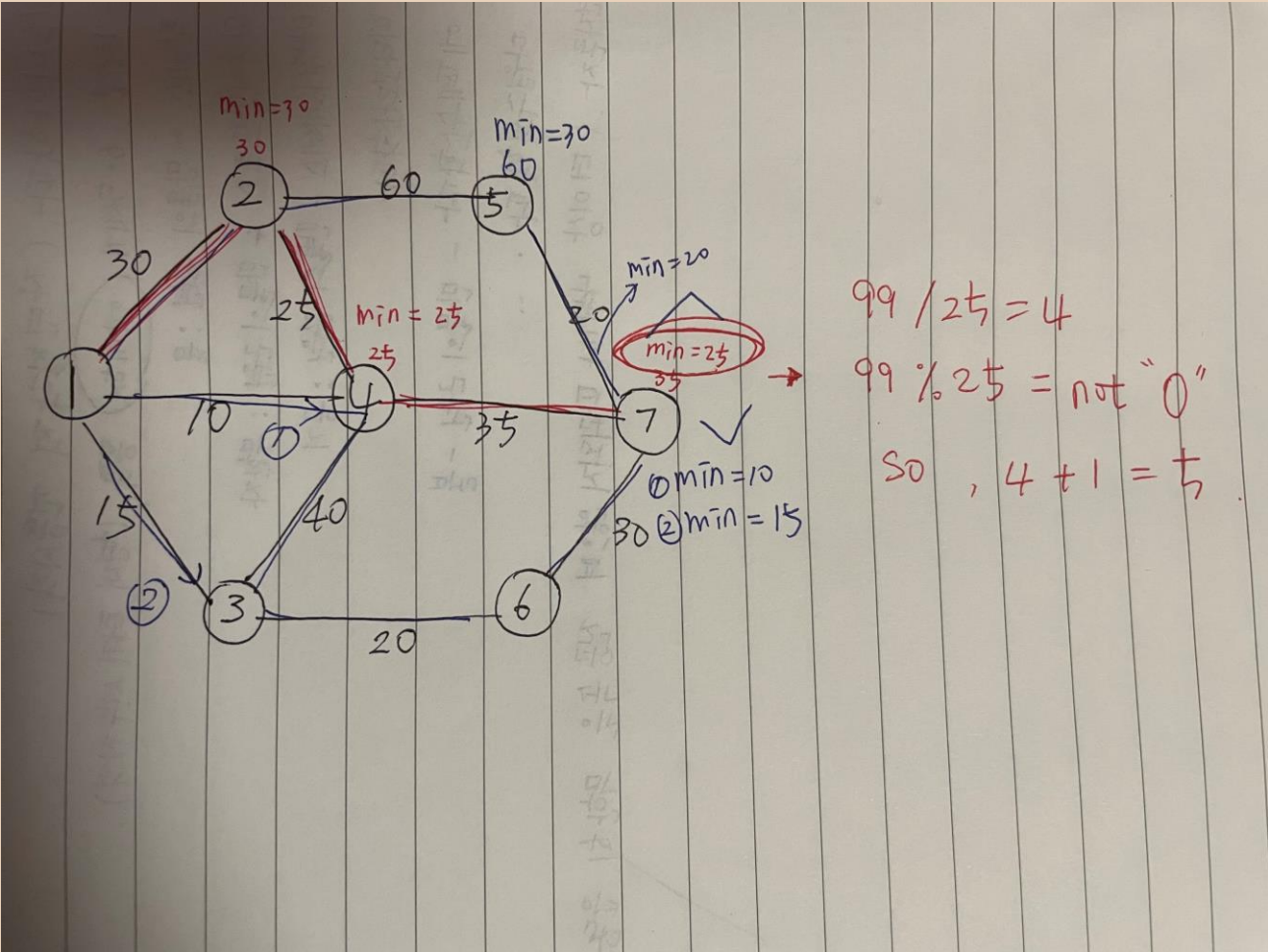
Microsoft Visual Studio 디버그 콘솔

```
7 10
1 2 30
1 3 15
1 4 10
2 4 25
2 5 60
3 4 40
3 6 20
4 7 35
5 7 20
6 7 30

1 7 99
Scenario #1
Minimum Number of Trips = 5

0 0

C:\Users\yeonhee\source\repos\Project\
이 창을 닫으려면 아무 키나 누르세요.
```



Thank you 😊