

ACM算法与程序设计

第二讲

简单计算题（一）

数学科学学院：汪小平
wxiaoping325@163.com



Fibonacci Again

<http://acm.uestc.edu.cn/#/problem/show/1008>





- **Problem Description**

There are another kind of Fibonacci numbers: $F(0) = 7$, $F(1) = 11$, $F(n) = F(n-1) + F(n-2)$ ($n \geq 2$).

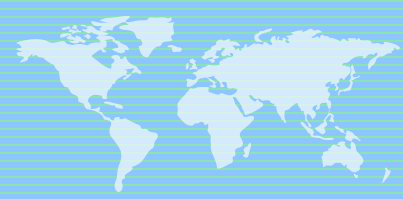
- **Input**

Input consists of a sequence of lines, each containing an integer n . ($n < 1,000,000$).

- **Output**

Print the word "yes" if 3 divide evenly into $F(n)$.

Print the word "no" if not.



- **Sample input:**

0 1 2 3 4 5 6

- **Sample output:**

no no yes no no no yes



题目分析

- 能被3整除的整数的特点？
- 如果两个数的和能被3整除，这两个数有什么特点？
- 关于能否被3整除，这两个数一共有多少种组合？

还要看程序吗？



程序清单:

```
#include<stdio.h>
int main()
{
    long n;
    while(scanf("%ld",&n) == 1)
        if (n%8==2 || n%8==6)
            printf("yes\n");
        else
            printf("no\n");
    return 0;
}
```



Rightmost Digit

<http://acm.uestc.edu.cn/#/problem/show/1009>

- **Problem Description**

Given a positive integer N , you should output the most right digit of N^N .

- **Input**

The input contains several test cases. The first line of the input is a single integer T which is the number of test cases. T test cases follow. Each test case contains a single positive integer N ($1 \leq N \leq 1,000,000,000$).

- **Output**

For each test case, you should output the rightmost digit of N^N .



- **Sample input:**

2

3

4

- **Sample output:**

7

6



- 数据规模 → 很大
- 暴力方法 → 该打
- 基本思路 → 规律

```
#include<stdio.h>
int main(void)
{
    int T,i,a,digit;
    int n;
    scanf("%d",&T);
    for(i=0;i<T;i++)
    {
        scanf("%d",&n);
        a=n%10;
        if(a==0 || a==1 || a==5
            || a==6 || a==9)
            digit=a;
        else if(a==2)
        {
            if(n%4==0) digit=6;
            else digit=4;
        }
    }
```

```
    else if(a==3)
    {
        if(n%4==1) digit=3;
        else digit=7;
    }
    else if(a==4)
        digit=6;
    else if(a==7)
    {
        if(n%4==1) digit=7;
        else digit=3;
    }
    else //a=8
    {
        if(n%4==0) digit=6;
        else digit=4;
    }
    printf("%d\n",digit);
}
return 0;}
```



偏僻的小路

1、Link

<http://acm.uestc.edu.cn/#/problem/show/154>

2、Description

在电子科大清水河校区的某个偏僻角落里，有一条东西方向的小路，长 L 米（由西向东位置为 0 到 L ），小路上有 N 个人从 $t=0$ 秒开始以相同的恒定速率 V 米/秒前进（面朝西或面朝东）。这条小路太偏僻了，所有人都想尽快离开这条小路。不幸的是，当两个人相遇时，只有男生会给女生让路（视为两人擦肩而过），男生遇上男生、女生遇上女生时，谁也不肯让路，只好都无奈的掉头往回走。

现在HS很好奇，想知道最后一个人离开小路的时间，以及所有人在小路上走的路程的总和，你能编写程序帮助他吗？



偏僻的小路

3、Input

第一行包括3个整数， N, L, V , 表示小路上的人数、小路的长度、所有人前进的速率 ($N \leq 100, L \leq 1000000, V > 0$)

接下来有 N 行，每行3个数据，第 i 行的数据表示第 i 个人的位置（从0到 L 的整数）、性别（M或F）、方向（W表示面朝西、E表示面朝东）

当 $N=L=V=0$ 时，输入结束

4、Output

对于每组输入，输出一行两个小数，表示最后一个人离开的时间以及所有人在小路上走的路程的总和，用一个空格隔开，答案四舍五入保留两位小数。



偏僻的小路

5、Sample Input

2 4 2

1 M E

3 M W

0 0 0

6、Sample Output

1.50 6.00

7、Source

royce



偏僻的小路

分析

```

#include <stdio.h>
int N, L, V, pos;
char sex[4], dir[4];
int main()
{
    double last, dist;//最后时间与路程
    double len;//当前人的路程
    while (scanf("%d %d %d", &N, &L, &V) == 3)
    {
        if (N == 0 && L == 0 && V == 0) break;
        last = 0.0, dist = 0.0;//清零
        for (int i = 0; i < N; i++)
        {
            scanf("%d %s %s", &pos, sex, dir);
            if (dir[0] == 'W') len = pos;
            else len = L - pos;
            if (len / V > last) last = len / V;
            dist += len;
        }
        printf("%.2f %.2f\n", last, dist);
    }
    return 0;
}

```



约会

1、Link

<http://acm.uestc.edu.cn/#/problem/show/156>

2、Description

有一天silentsky和lcy同学去教室上自习。silentsky百无聊赖地看着书本，觉得很无聊，看着右手边的lcy认真仔细的在画着她繁重的物理实验报告的图。silentsky无聊地弄着他的脉动瓶子，结果一不小心就把瓶盖弄到了lcy刚画好的坐标纸上，而且冥冥之中仿佛有一双手在安排，瓶盖的中心正好和坐标纸的中心重合了，瓶盖的边缘有水，会弄湿坐标纸的。lcy很生气，后果很严重。

于是，lcy由此情形想出了一道难题问silentsky，如果他回答正确了。lcy就原谅了silentsky并且答应他星期天去看暮光之城2的请求，不然一切都免谈。然后silentsky就回去面壁思过了，现在silentsky好无助的，希望得到广大编程爱好者的热心帮助。

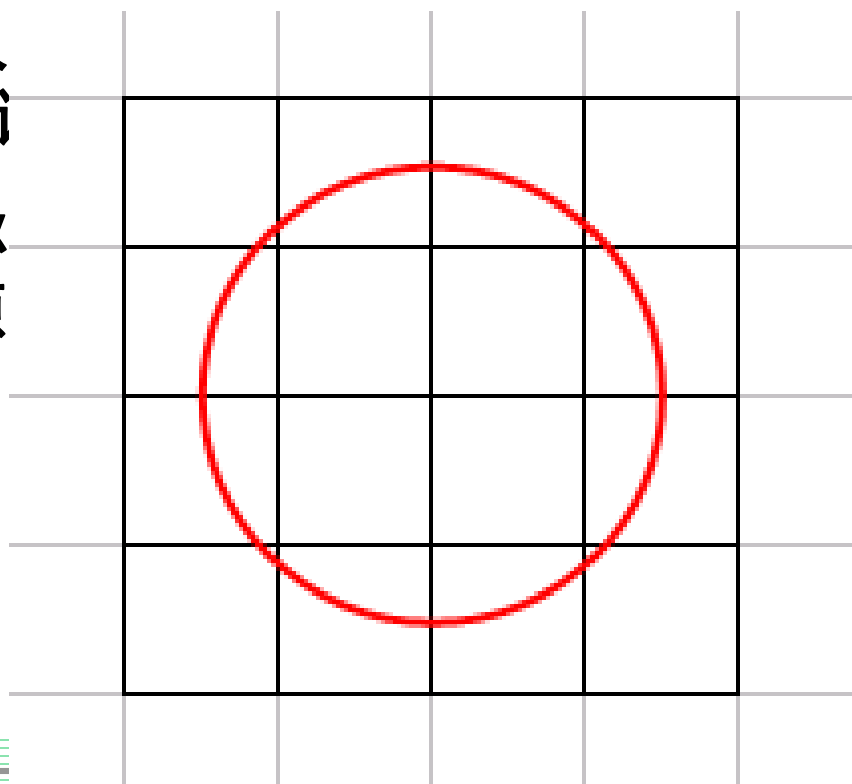


约会

问题是这样的: lcy现在手上有一张 $2n * 2n$ 的坐标纸, 而silentsky的圆形瓶盖的直径正好有 $2 * n - 1$ 大, 现在lcy想知道silentsky到底弄湿了多少个坐标纸的格子 (坐标纸是由 $1 * 1$ 的小格子组成的表格)

如果还是有人
silentsky做下翻i

问题就是给你
然后以中心为原
了多少个格子。



勺意思。干脆
y的 $O(n_n)O\sim$ 。

成 $1 * 1$ 的格子,
圆的周线穿过



约会

3、Input

含有多组测试数据，每组数据都包含一个正整数 n （ $n \leq 1000$ ）。

当 $n = 0$ 的时候结束程序，证明silentky经受住考验了的 $O(n \cdot n)$ 。

4、Output

对于每个 n ，输出被瓶盖边缘的水弄湿了的格子数为多少。



约会

5、Sample Input

1

2

0

6、Sample Output

4

12

7、Source

love8909 & silentsky



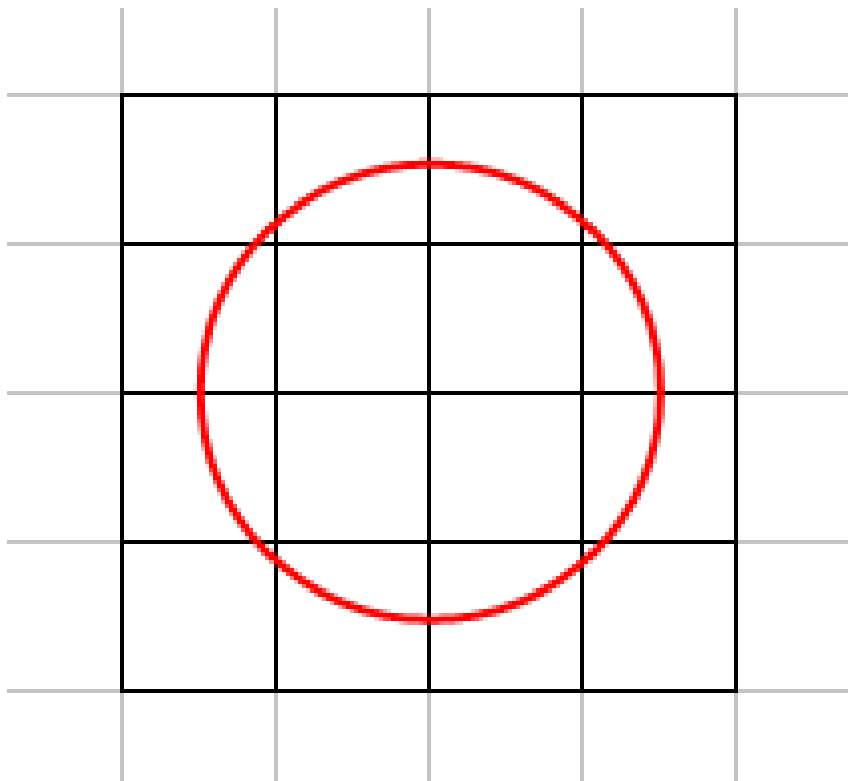
约会

分析

分段思想

1、直线相交

2、圆相交





约会

```
#include<stdio.h>
int main()
{
    int n;
    while(scanf("%d", &n) == 1)
    {
        if(n == 0)
            break;
        printf("%d\n", n * 8 - 4);
    }
    return 0;
}
```



木杆上的蚂蚁

<http://acm.uestc.edu.cn/#/problem/show/300>

Description

在一根细木杆上，有一些速度相同蚂蚁，它们有的往左走，有的往右走，木杆很细，只允许一只蚂蚁通过，所以当两只蚂蚁碰头的时候，它们会掉头继续前进，直到走出边界，掉下木杆。

已知木杆的长度和每只蚂蚁的名字、位置和初始方向，问依次掉下木杆的蚂蚁花费的时间以及它的名字。



Input

输入包含多组测试数据。

第一行包含一个整数 T ($T \leq 20$), 代表测试数据组数。

每组测试数据的第一行包含两个整数 N L , 表示有 N 只蚂蚁 ($N \leq 100$), 木杆长度为 L ($L \leq 1000$)。假设蚂蚁每秒前进一个单位距离, 掉头转向的时间忽略不计。

以下 N 行, 每行依次为蚂蚁的名字 (长度不超过10, 仅由英文字母组成), 初始位置 p ($0 < p < L$, 整数, 表示蚂蚁离木杆最左端的距离), 初始方向 (一个字符, L表示向左, R表示向右), 以单个空格分隔, 数据保证初始不会有两只蚂蚁在同一个位置。



Output

对于第k组测试数据，首先输出一行为“Case #k:”。

然后输出N行，给出依次掉下木杆的蚂蚁花费的时间以及它的名字，以单个空格分隔。

（按照掉下木杆的先后顺序输出，数据保证不会有两只蚂蚁同时掉下木杆）。

Sample Input

```
2
2 5
GG 1 L
MM 3 R
2 5
GG 1 R
MM 2 L
```

Sample Output

```
Case #1:
1 GG
2 MM
Case #2:
2 GG
4 MM
```



```
#include <cstdio>
#include <algorithm> //因为要用sort算法
#define N 100
using namespace std; //必须引用名字空间std
struct ant_type
{
    int pos;
    char name[11];
} ants[N];
struct event_type
{
    int drop_time;
    char dir;
} events[N];
bool cmp_ant(const ant_type& p, const ant_type& q)
{
    return p.pos < q.pos;
}
```

```
bool cmp_event(const event_type& p, const event_type& q)
{   return p.drop_time < q.drop_time;
}
```

```
int main()
```

```
{
```

```
    char dir[2];
```

```
    int i, k, n, L, R, T;
```

```
    scanf("%d", &T);
```

```
    for (k = 1; k <= T; k++)
```

```
    {
```

```
        scanf("%d%d", &n, &L);
```

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

```
        {
```

```
            scanf("%s%d%s", ants[i].name, &ants[i].pos, dir);
```

```
            events[i].dir = dir[0];
```

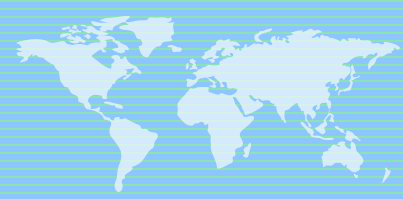
```
            events[i].drop_time = (dir[0] == 'L' ?
```

```
                ants[i].pos : L - ants[i].pos);
```

```
        }
```

```
sort(ants, ants + n, cmp_ant);
sort(events, events + n, cmp_event);
printf("Case #%-d:\n", k);
L = 0;    R = n - 1;
for (i = 0; i < n; i++)
{
    if (events[i].dir == 'L')
    {
        printf("%d %s\n", events[i].drop_time, ants[L].name);
        L++;
    }
    else
    {
        printf("%d %s\n", events[i].drop_time, ants[R].name);
        R--;
    }
}
return 0;
```

```
}
```



Flagstone

<http://acm.uestc.edu.cn/#/problem/show/50>

- **Description**

In the Qingshuihe Campus of UESTC, the most annoy problem to students are the flagstone path on the lawn. The designer seems so stupid that the flagstone path often make students step in the gap. Now a perfect step is wanted in order to not step in any gaps and step on every flagstone. The step length is required to be constant while the length of the flagstone and gap are given different.

The problem is asking you to tell the minimum length of the perfect step. To simplify the question, the foot is considered to be a point and the very beginning is the fore edge of the first flagstone, which also means the first flagstone has already been stepped on.



- **Input**

The first line of the input contains one integer T , which indicate the number of test cases. In each test case, the first line contains an integer N ($2 \leq N \leq 1e5$), indicating the number of flagstone. Following N lines, and each line is the length of one flagstone. And the following $N-1$ lines are the length of the gaps. All data is integer. All the length will be a positive integer, and the sum of them will fit in a 32bit signed integer.

- **Output**

One line for each test case contains only one number indicating the answer. One real number indicating the perfect step length should be accurate to two digits after the radix point. If it is impossible to find out a perfect step, just output “impossible” !



- **Sample Input**

2

2

10

20

5

3

10

20

5

5

1000

- **Sample Output**

15.00

impossible



- 每个石板踏且仅踏一次
- 对于第 i 块石板，一定是踏了 $i - 1$ 步到达的。
- 设第 i 块石板的左边界和右边界离起点的距离分别为 L 和 R ，可以确定步长必须在区间 $[L / (i - 1), R / (i - 1)]$ 之内。
- 问题转化为求多个区间的交。如果交集为空，则答案为 impossible，否则输出交区间的左界。



```
#include<stdio.h>
```

```
int a[100001];
```

```
int b[100001];
```

```
int main()
```

```
{
```

```
    int t,p;
```

```
    int n,i;
```

```
    double h,l;
```

```
    double hh,ll;
```

```
    double s;
```

```
    scanf("%d",&t);
```

```
    for (p=1;p<=t;p++)
```

```
    {
```


```
        scanf("%d",&n);
```

```
        for (i=1;i<=n;i++)
```

```
            scanf("%d",&a[i]);
```

```
        for (i=1;i<n;i++)
```

```
            scanf("%d",&b[i]);
```

```
s=0;
l=a[1]+b[1];
h=a[1]+b[1]+a[2];
for (i=1;i<n;i++)
{
    s=s+a[i]+b[i];
    ll=s/(double)i;
    hh=(s+a[i+1])/(double)i;
    if (ll>l) l=ll;
    if (hh<h) h=hh;
}
if (l<=h) printf("%.2lf\n",l);
else printf("impossible\n");
}
return 0;
}
```



CDOJ 输出前m大的数据

<http://acm.uestc.edu.cn/#/problem/show/528>

- **Problem Description**

给你n个整数，请按从大到小的顺序输出其中前m大的数。

- **Input**

每组测试数据有两行，第一行有两个数n,m($0 < n, m < 1000000$), 第二行包含n个各不相同, 且都处于区间 $[-500000, 500000]$ 的整数。

- **Output**

对每组测试数据按从大到小的顺序输出前m大的数。



- **Sample input:**

5 3

3 -35 92 213 -644

- **Sample output:**

213 92 3



- 常规的思想是？
 - 常规的结果是？
 - 数据的特点是？
 - 加速的方法是？
 - 如果数据可以重复呢？
- 用最好的排序
 - TLE
 - 各不相同
 - HASH
 - 处理冲突

```

#include <stdio.h>
#include <string.h>
#define N 1000000
int a[N];
int main(void)
{
    int i,j,n,m,t;
    while(scanf("%d%d",&n,&m)==2)
    {
        memset(a,0,N*sizeof(int));
        for(i=0;i<n;i++)
        {
            scanf("%d",&t);
            a[t+N/2]++; //下标对应数+N/2, 元素值存储重复次数
        }
        for(t=0,i=N-1;t<m && i>=0;)
        {
            if(a[i]>0)//对应数
            {
                printf("%d ",i-N/2); //打印对应的数
                t++; //计数器增加
                if(--a[i]==0) i--;
            }
            else i--;
        }
        printf("\n");
    }
    return 0;
}

```



猴子选大王

<http://acm.uestc.edu.cn/#/problem/show/525>

有 m 个猴子围成一圈，按顺时针编号，分别为1到 m 。现打算从中选出一个大王。经过协商，决定选大王的规则如下：从第一个开始顺时针报数，报到 n 的猴子出圈，紧接着从下一个又从1顺时针循环报数，...，如此下去，最后剩下下来的就是大王。

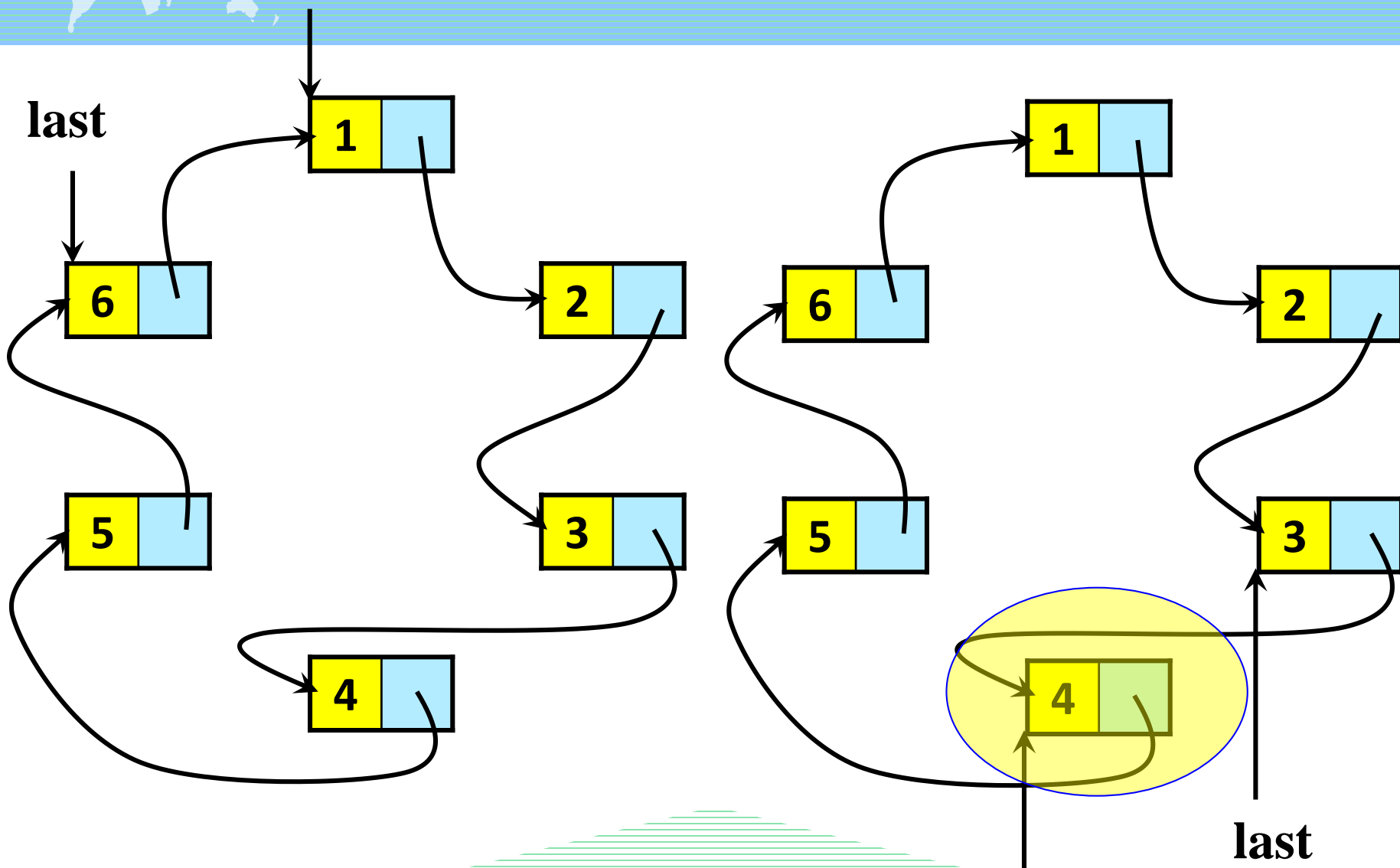
Sample Input	Sample Output
1 3 2	3



pre

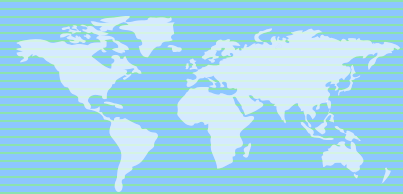
$m=6, n=4$

last

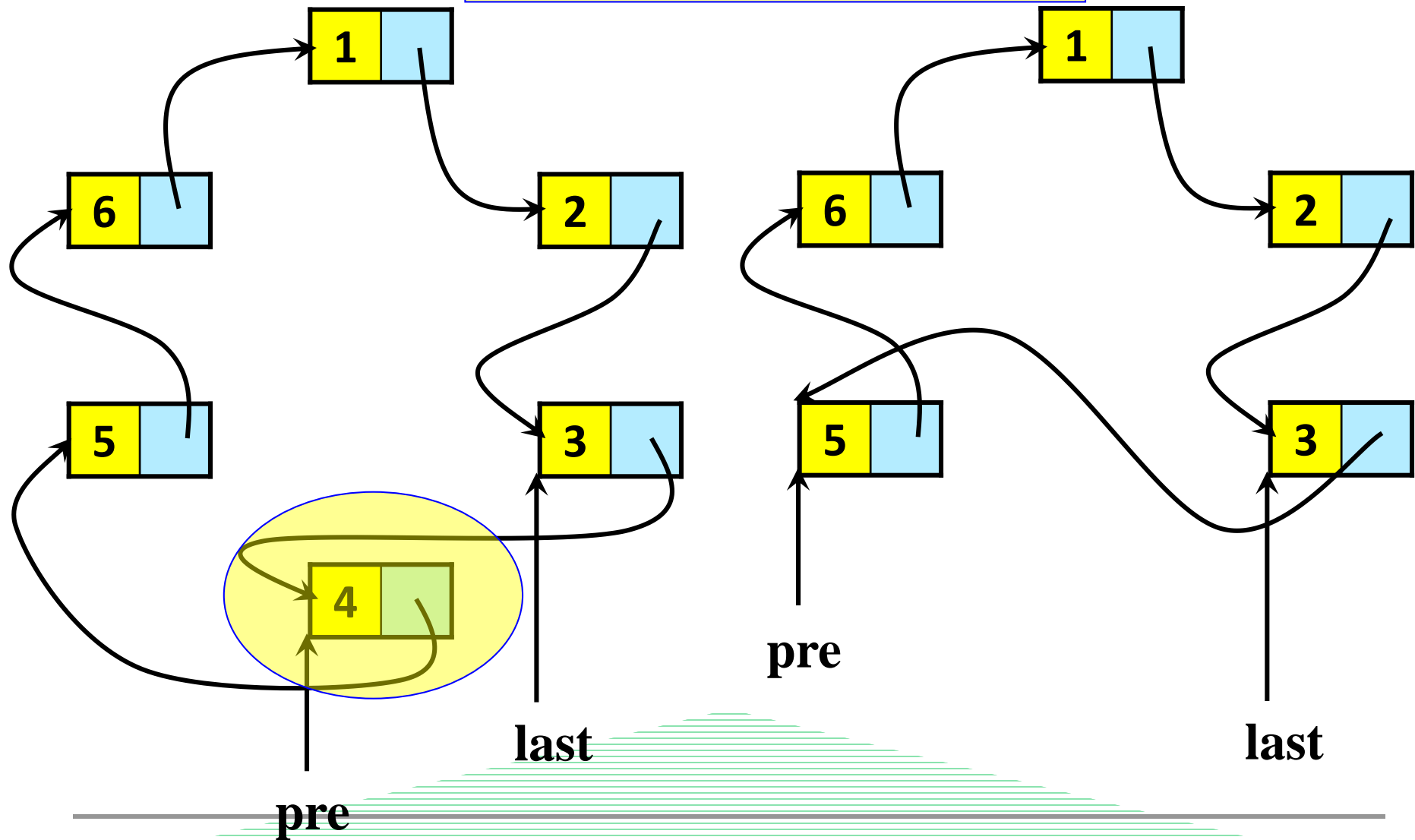


pre

last



```
last->next = pre->next;  
free(pre);  
pre = last->next;
```




```
#include <stdio.h>
typedef struct _monkey
{
    int number;
    struct _monkey *next;
} monkey;
int main()
{
    int T;
    scanf("%d", &T);
    while(T--)
    {
        int i, m, n, counter;
        monkey *pre, *last, *newnode;
        scanf("%d%d", &m, &n);
        pre = (monkey*)malloc(sizeof(monkey));
        pre->number = 1;    last = pre;
```

```
for(i = 2; i <= m; i++)
{
    newnode = (monkey*)malloc(sizeof(monkey));
    newnode->number = i;      last->next = newnode;
    last = newnode;
}
last->next = pre;//循环链表建立完毕
for(i = 0; i < m-1; i++)//每次退出一次猴子
{
    counter = 1;
    while(counter < n)
    {
        last = pre;    pre = pre->next;    counter++;
    }
    last->next = pre->next;  free(pre);    pre = last->next;
}
printf("%d\n", pre->number);
free(pre);
}
return 0;
}
```

```
#include <stdio.h>
#define N 103
//首先考虑猴子的编号为0 ~ m-1, 输出时+1即可
int main(void)
{
    int T;
    scanf("%d", &T);
    while(T--)
    {
        int i, m, n, counter;
        int last, pre, monkey[N];
        scanf("%d%d", &m, &n);
        for(i = 0; i < m; i++) //形成静态链表
            monkey[i] = (i + 1) % m;
        last = m-1;
        pre = 0; //第一次报数猴子的下标
```

```
for(i = 0; i < m - 1; i++) //每次退出一个
{
    counter = 1;
    while(counter < n)
    {
        last = pre; //保留上一个的下标
        pre = monkey[pre]; //得到下一个猴子的下标
        counter++;
    }
    monkey[last] = monkey[pre]; //改变链接，退出，
    pre = monkey[pre]; //下一次第一个报数的猴子下标
}
printf("%d\n", pre + 1);
}
return 0;
}
```



最短路

<http://acm.uestc.edu.cn/#/problem/show/30>

- **Description**

在每年的校赛里，所有进入决赛的同学都会获得一件很漂亮的T-shirt。但是每当我们的工作人员把上百件的衣服从商店运回到赛场的时候，却是非常累的！所以现在他们想要寻找最短的从商店到赛场的路线，你可以帮助他们吗？



• Input

输入包括多组数据。每组数据第一行是两个整数 N 、 M ($N \leq 100$, $M \leq 10000$)， N 表示成都的大街上有几个路口，标号为1的路口是商店所在地，标号为 N 的路口是赛场所在地， M 则表示在成都有几条路。 $N=M=0$ 表示输入结束。接下来 M 行，每行包括3个整数 A , B , C ($1 \leq A, B \leq N, 1 \leq C \leq 1000$)，表示在路口 A 与路口 B 之间有一条路，我们的工作人员需要 C 分钟的时间走过这条路。输入保证至少存在1条商店到赛场的路线。

• Output

对于每组输入，输出一行，表示工作人员从商店走到赛场的最短时间



- **Sample Input**

2 1

1 2 3

3 3

1 2 5

2 3 5

3 1 2

0 0

- **Sample Output**

3

2



Dijkstra算法

Dijkstra算法能求一个顶点到另一顶点最短路径。它是由Dijkstra于1959年提出的。实际它能出始点到其它所有顶点的最短路径。

Dijkstra算法是一种标号法：给赋权图的每一个顶点记一个数，称为顶点的标号（临时标号，称T标号，或者固定标号，称为P标号）。T标号表示从始顶点到该标点的最短路长的上界；P标号则是从始顶点到该顶点的最短路长。

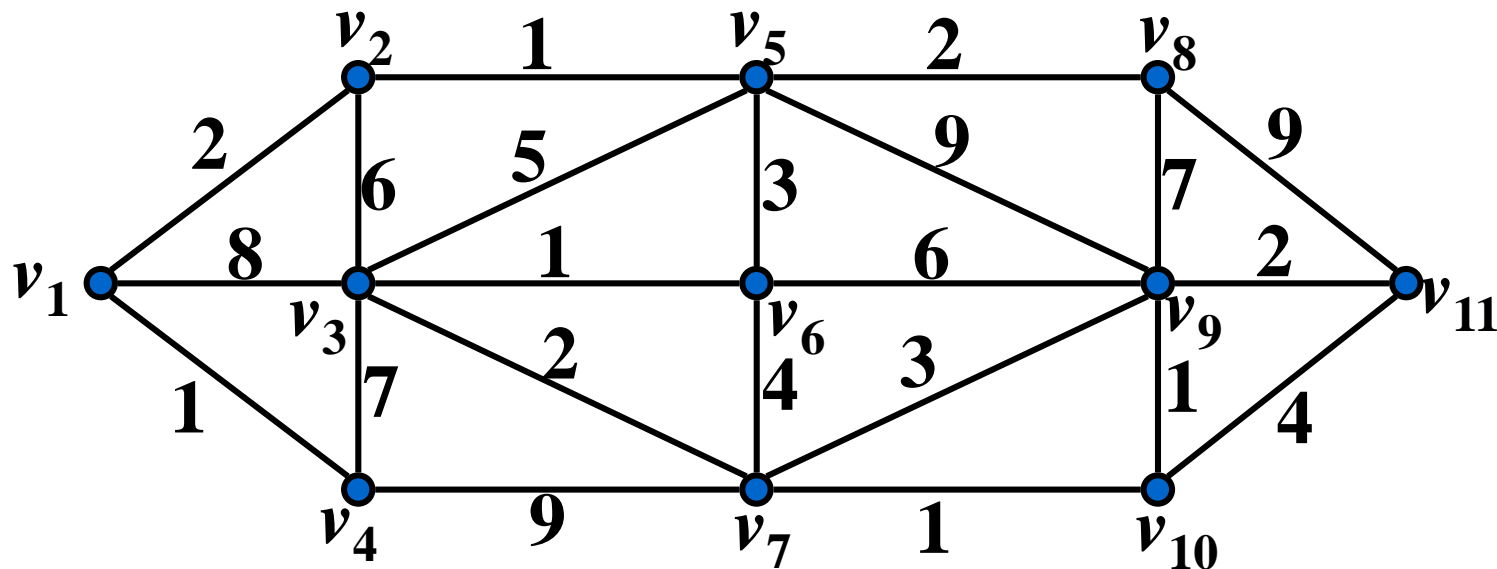
Dijkstra算法步骤如下：

Dijkstra算法

- (1) 给顶点 v_1 标P标号 $d(v_1) = 0$, 给顶点 $v_j (j = 2, 3, \dots, n)$ 标T标号 $d(v_j) = l_{1j}$;
- (2) 在所有T标号中取最小值, 譬如, $d(v_{j_0}) = l_{1j_0}$, 则把 v_{j_0} 的T标号改为P标号, 并重新计算具有T标号的其它各顶点的T标号: 选顶点 v_j 的T标号 $d(v_j)$ 与 $d(v_{j_0}) + l_{j_0j}$ 中较小者作为 v_j 的新的T标号。
- (3) 重复上述步骤, 直到目标顶点的标号改为P标号。

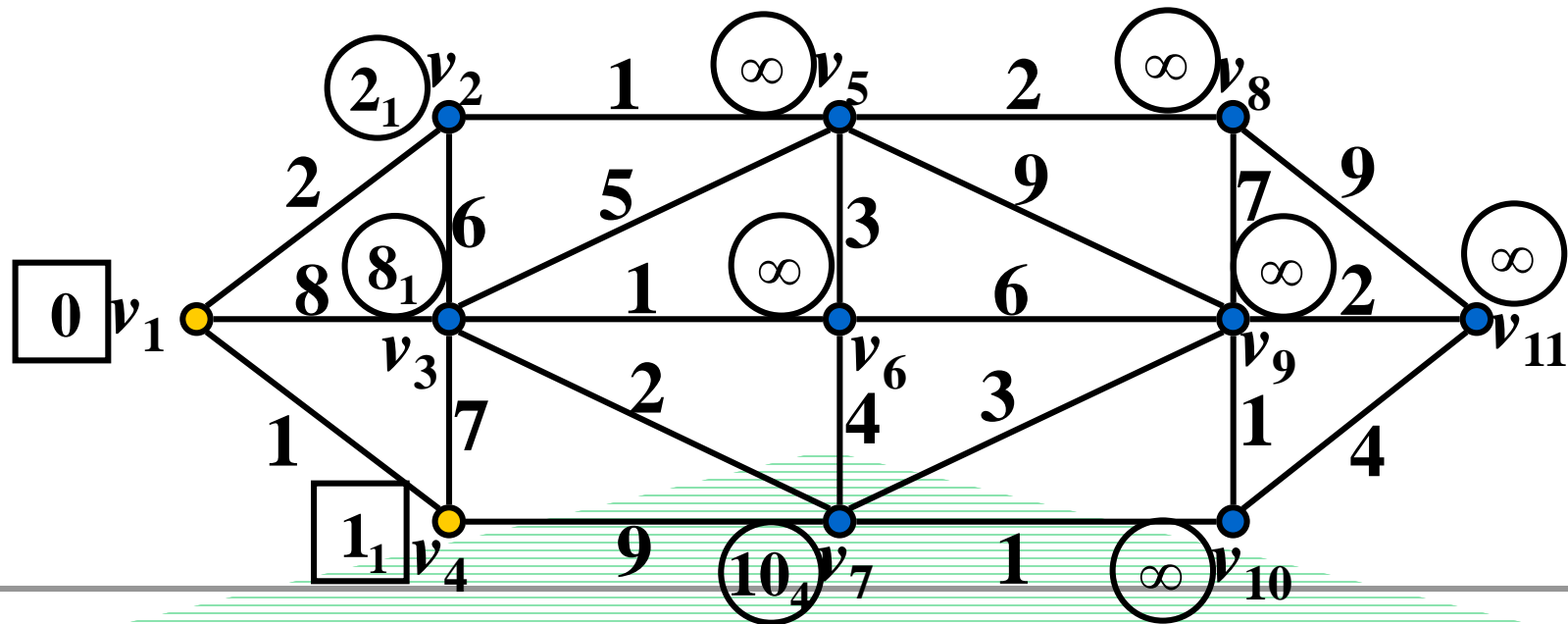
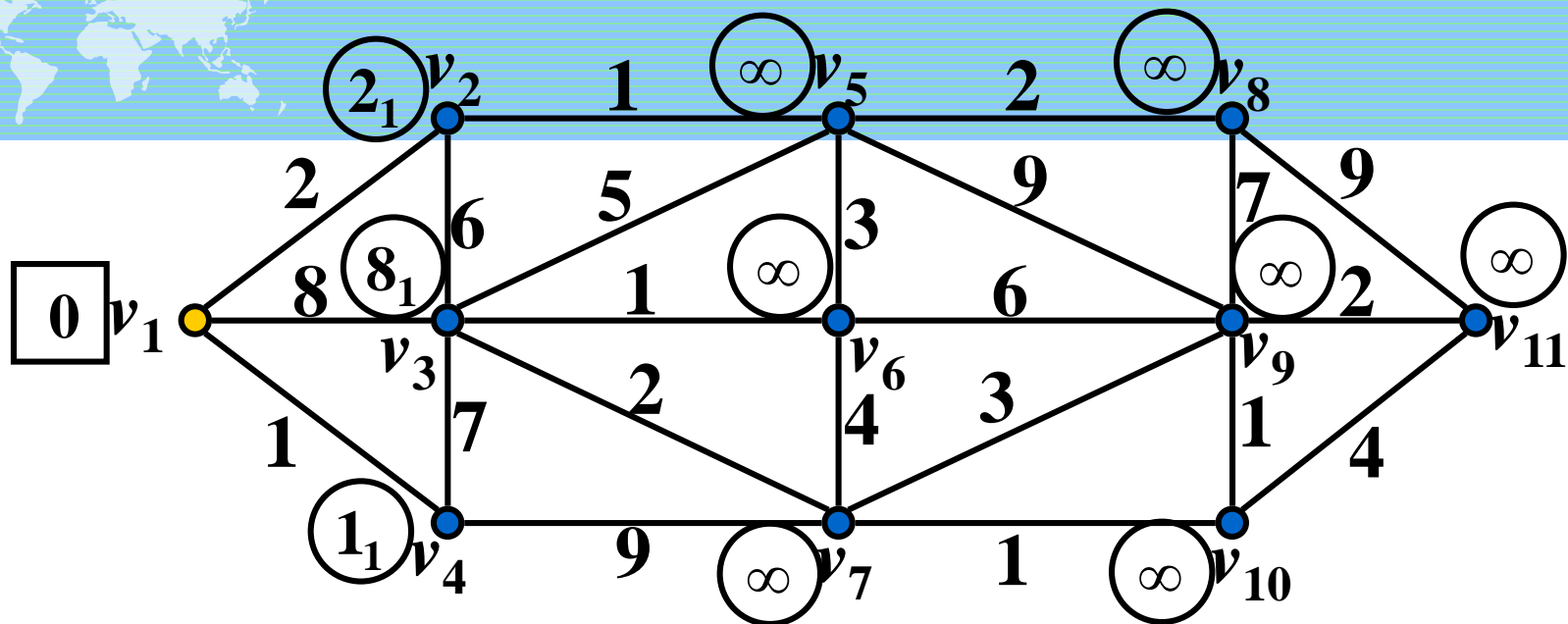


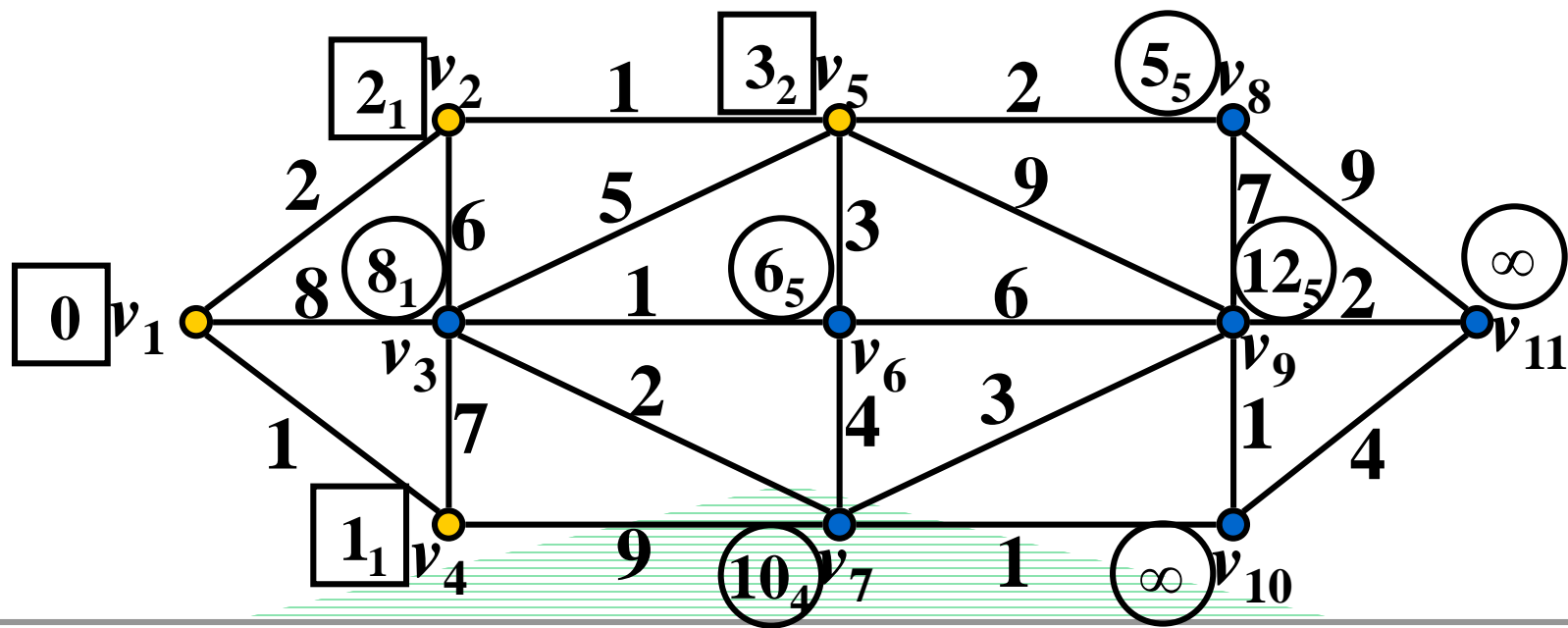
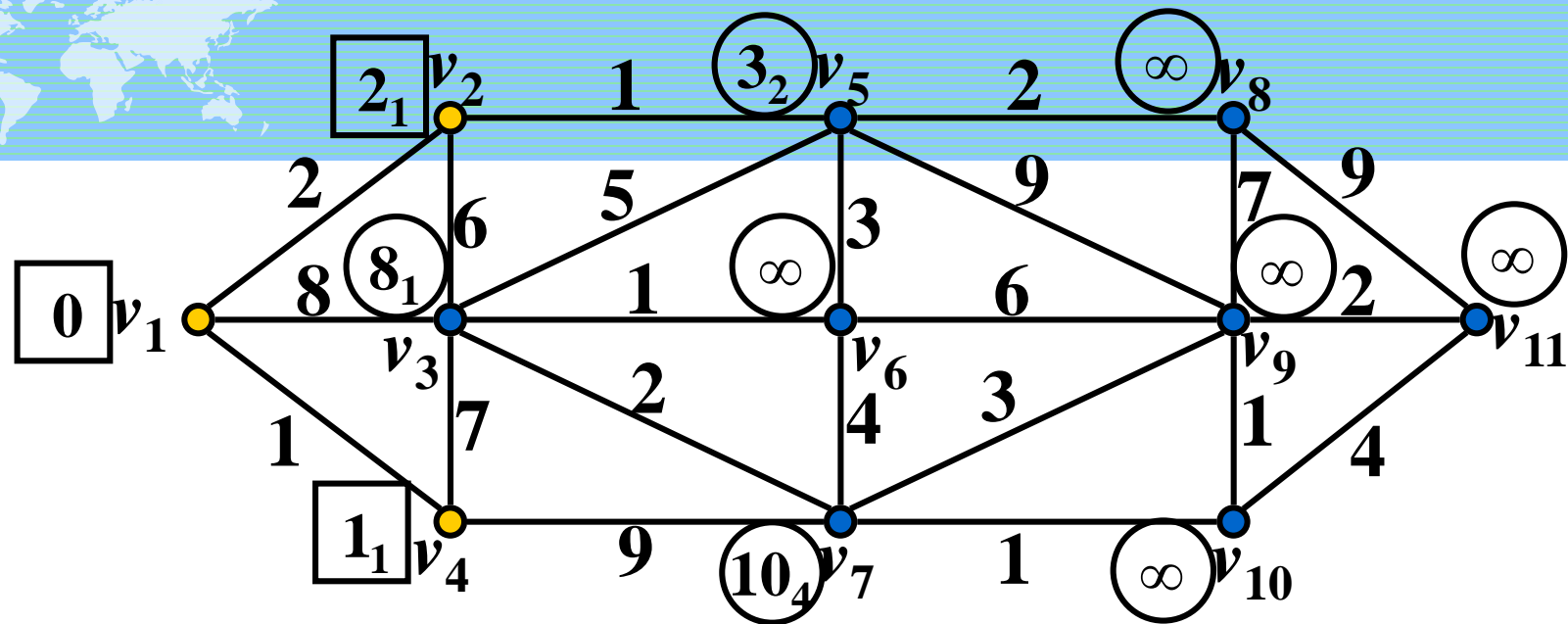
例 求出下图 G 中 v_1 到 v_{11} 的最短路长。

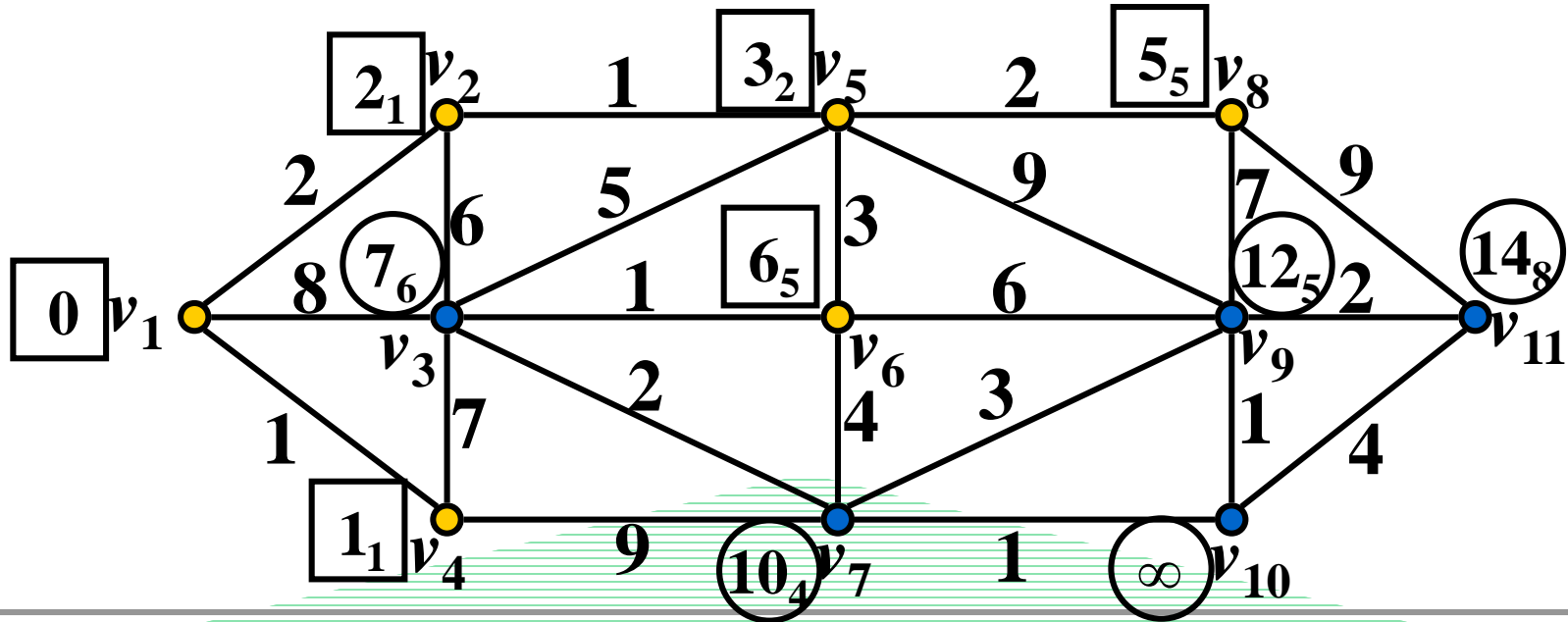
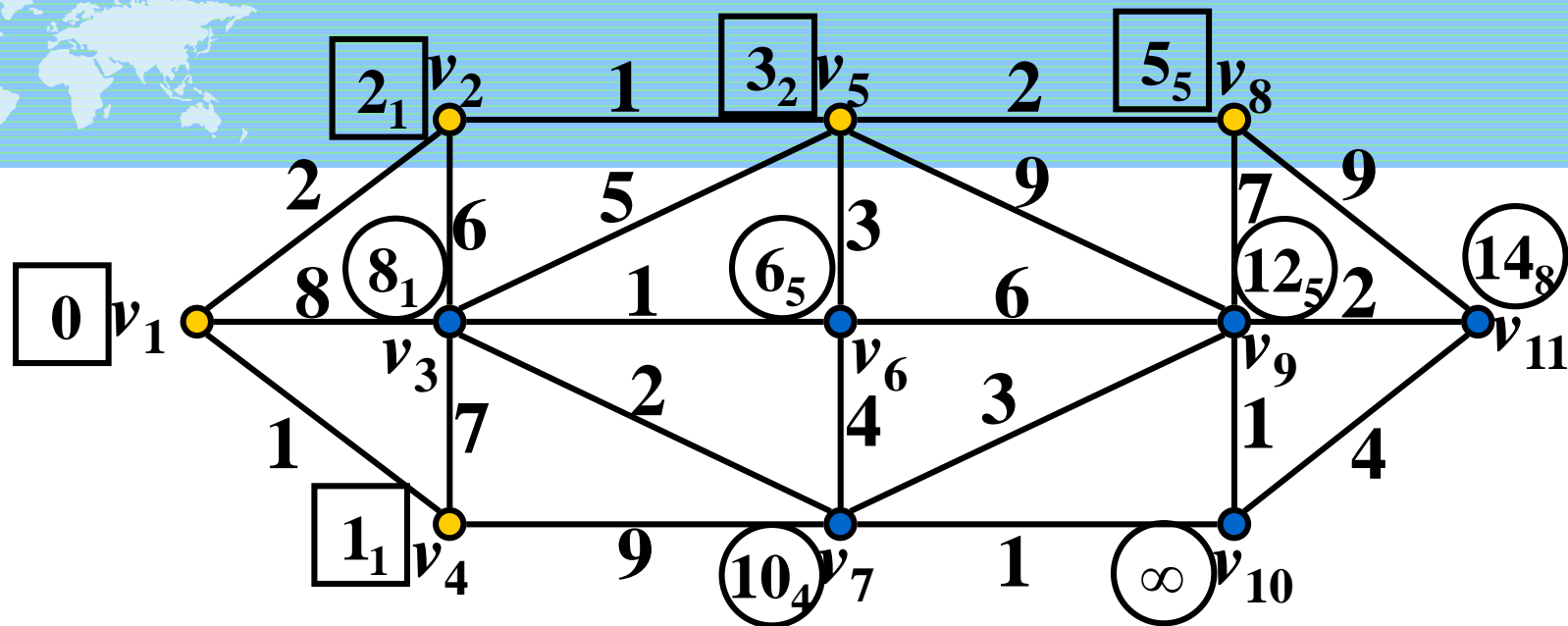


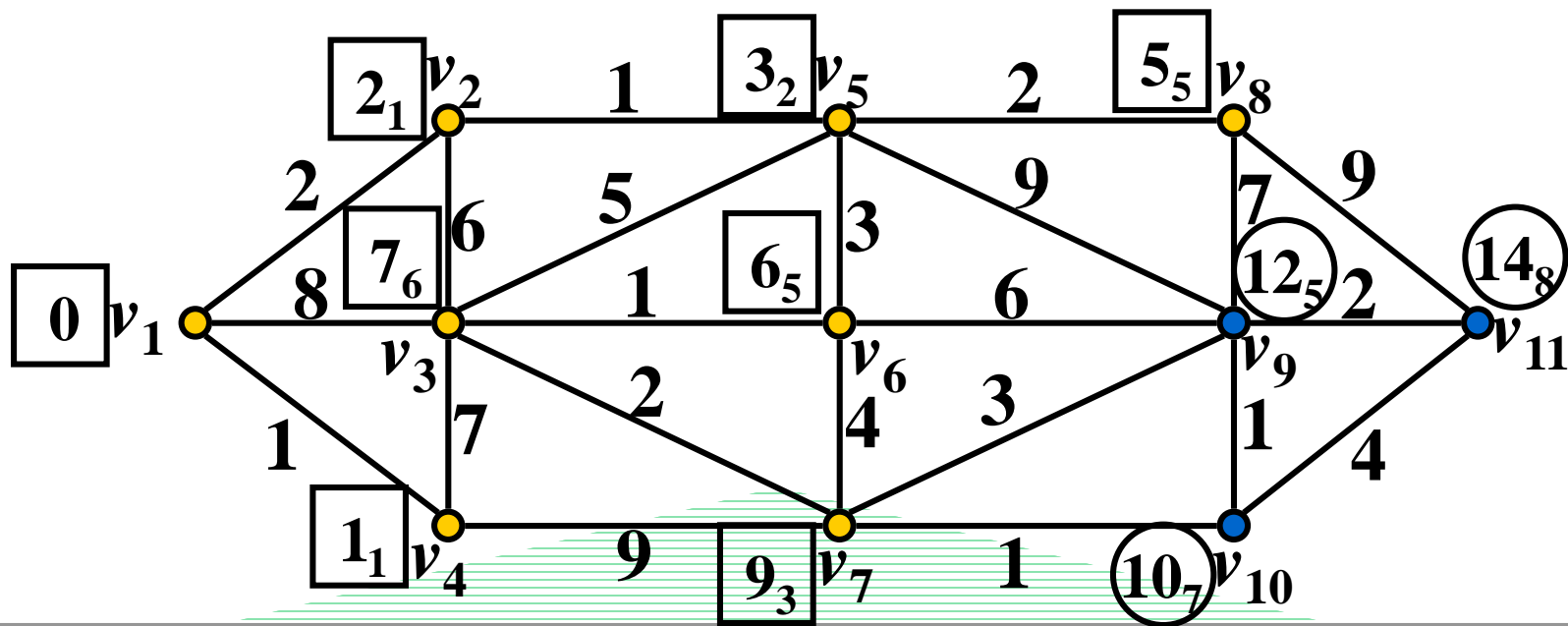
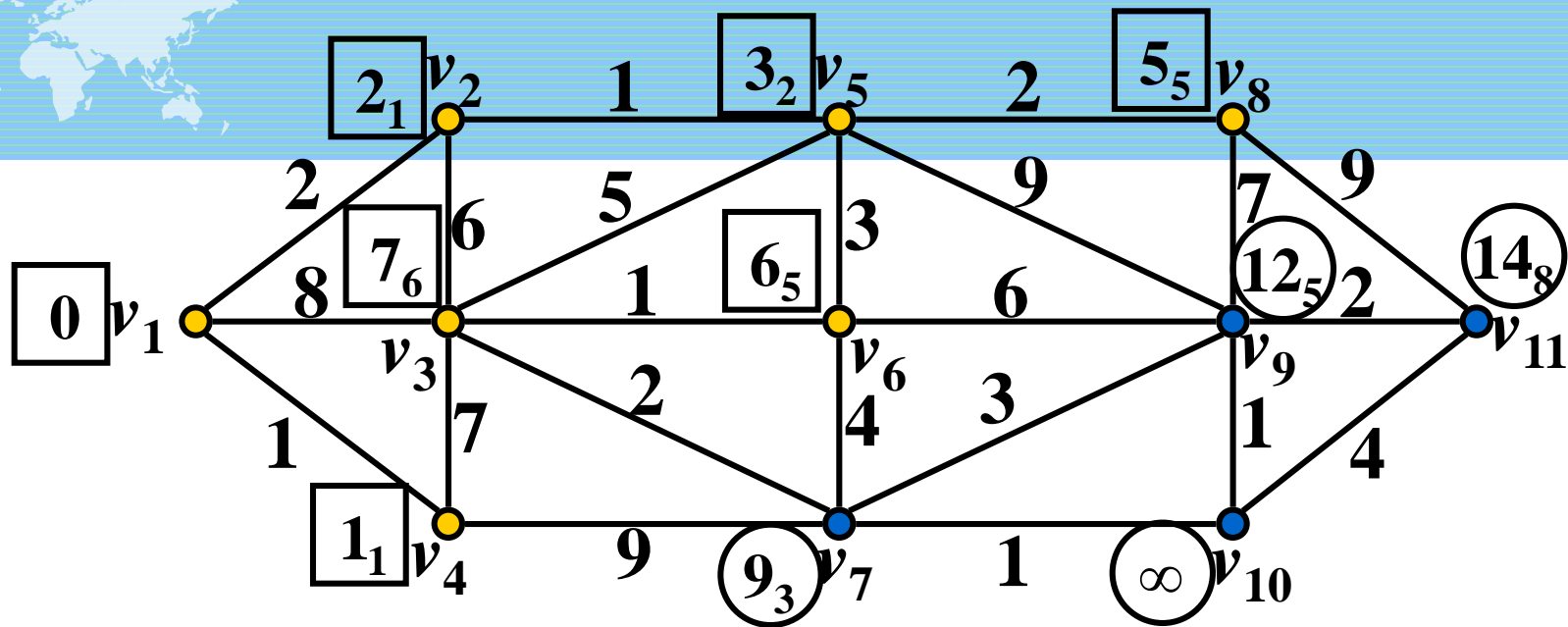
$\boxed{3_2}$ --- 固定编号，下标表示前趋顶点

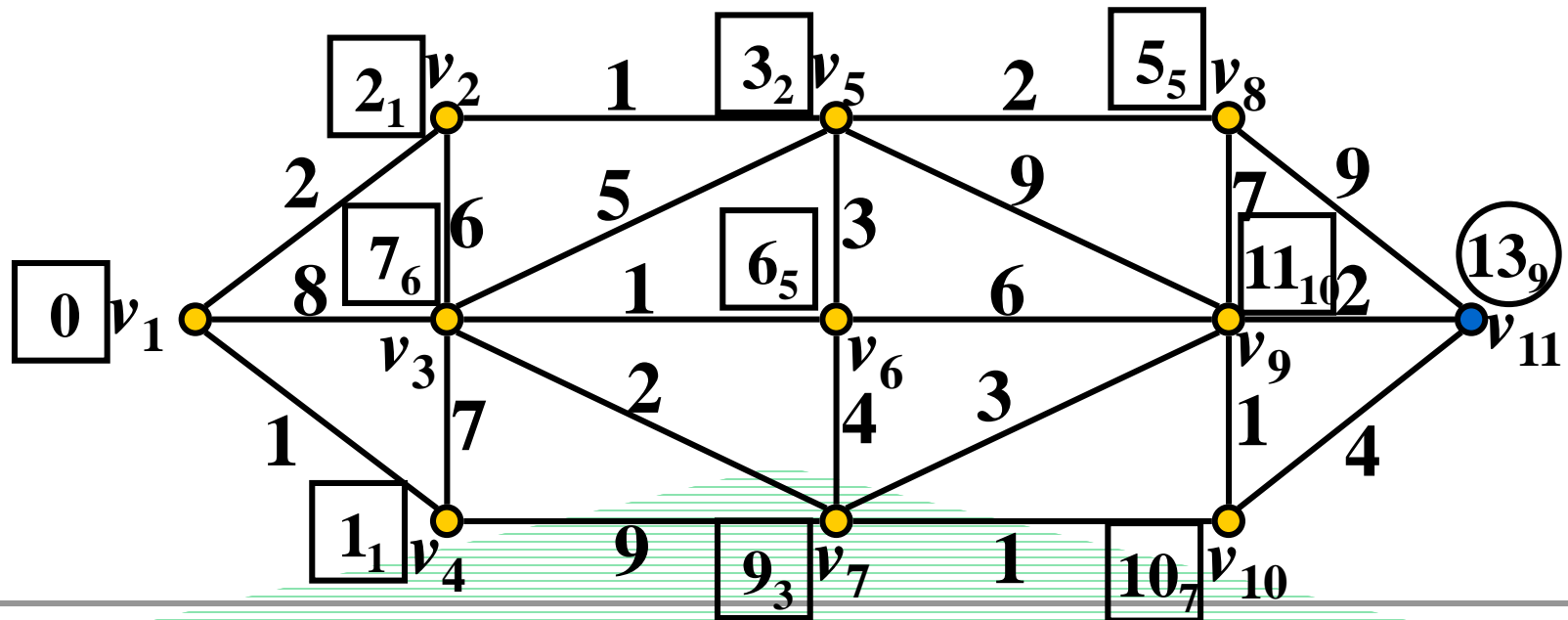
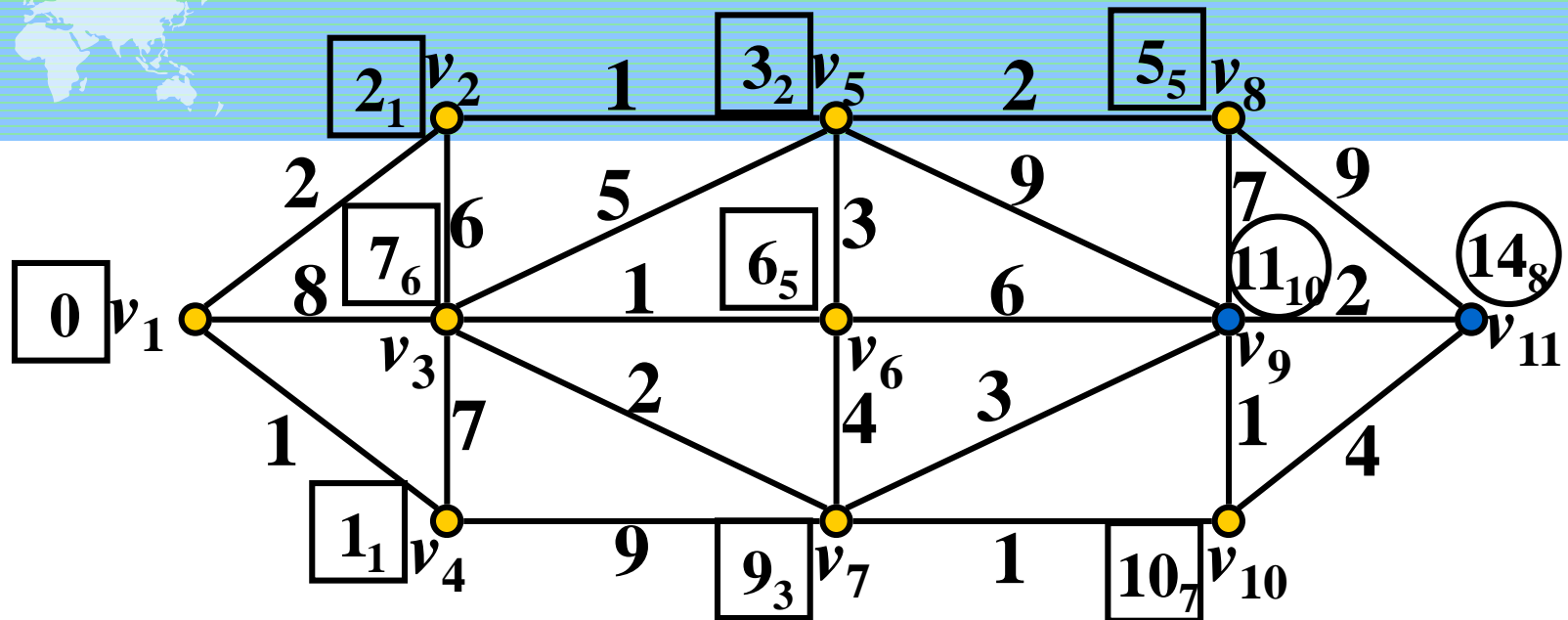
$\textcircled{8_1}$ --- 临时编号，下标表示前趋顶点

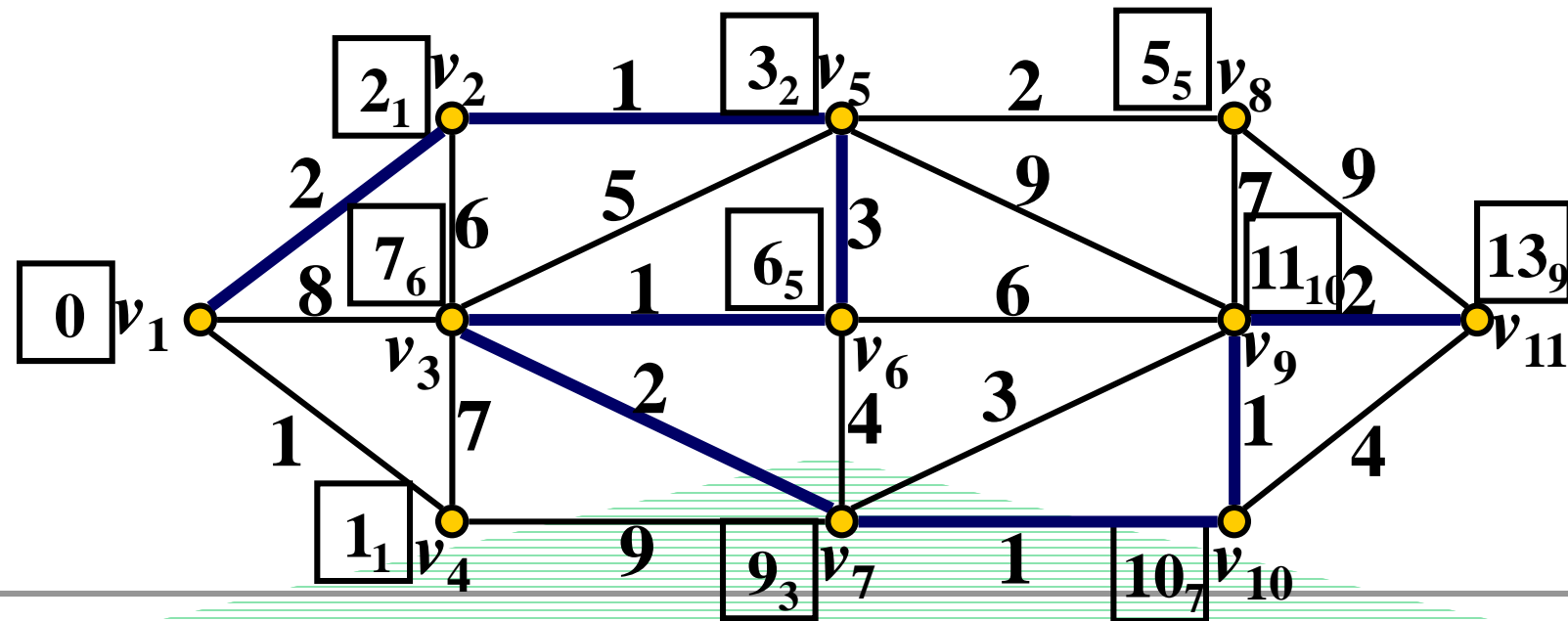
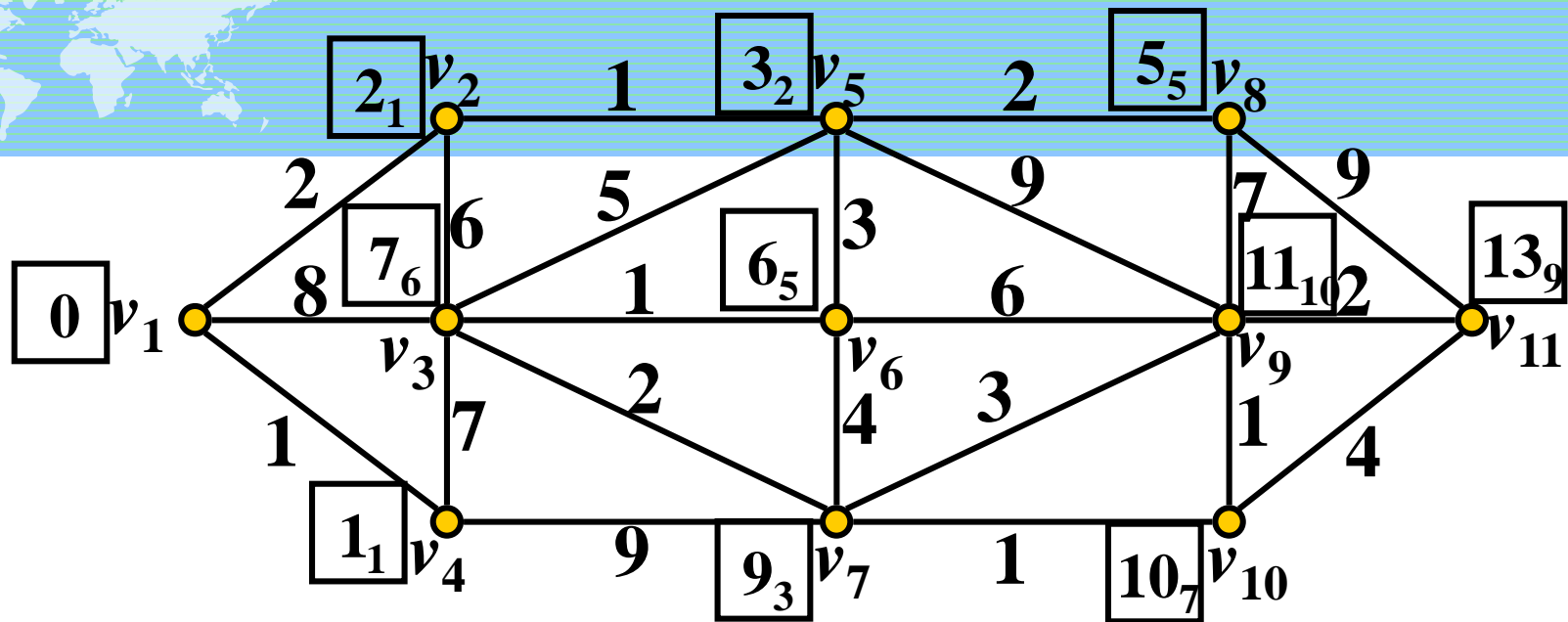
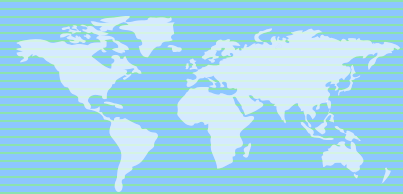













```

#include <stdio.h>
#include <string.h>
#define N 102
#define INF 10000000 //无穷
int map[N][N]; //邻接矩阵
int main()
{
    int a, b, c, i, j, n, m;
    int dis[N]; //记录起点到该点的最短距离
    int f[N]; //记录固定标号与临时标号
    while(scanf("%d%d", &n, &m) == 2)
    {
        if(n == 0 && m == 0) break;
        for (i = 0; i < N; i++)
            for (j = 0; j < N; j++)
                map[i][j] = INF;
        for (i = 0; i < m; i++)
        {
            scanf("%d%d%d", &a, &b, &c);
            a--; b--; //结点编号映射到0~n-1
            map[a][b] = map[b][a] = c; //无向图
        }
        for (i = 1; i < n; i++) dis[i] = map[0][i];
        memset(f, 0, sizeof(f));
        dis[0] = 0;
        f[0] = 1; //初始只有起点为固定标号
    }
}

```

```

for (i = 0; i < n-1; i++)
{
    a = -1; //临时编号中值最小的编号
    b = INF; //找临时编号中最小值
    for (j = 1; j < n; j++)
    {
        if (f[j] == 0 && dis[j] < b)
        {
            b = dis[j];
            a = j;
        }
    }
    if (b == -1) break; //说明无路到终点
    if (a == n-1) break; //已找到到终点的最短路
    f[a] = 1; //改为固定编号
    for (j = 1; j < n; j++) //修改相邻结点的标号值
        if ((f[j] == 0) && (b + map[a][j] < dis[j]))
            dis[j] = b + map[a][j];
}
if(dis[n-1] >= INF) printf("Impossible\n");
else printf("%d\n",dis[n-1]);
}
return 0;
}

```