

The Greedy Method

Idea

Xác định một trật tự xử lý để thu được kết quả tốt nhất.

Trật tự có thể là:

- Lấy min
- Lấy max
- Đan xen min max
- ...

Problem 1. Máy phát nhạc

Bảng nhạc ghi tuần tự n bản nhạc mã số $0..n-1$ với thời lượng phát khác nhau t_i . Mỗi ngày tần suất phát mỗi bản nhạc coi như bằng nhau. Để phát bản nhạc thứ i máy bỏ qua $i-1$ bài đầu tiên rồi phát bài i . Phát xong mỗi bài đầu đọc chuyển về đầu băng nhạc. Cần ghi các bài để tổng thời gian hoạt động của máy là nhỏ nhất.

Cách ghi 1

Bài	0	1	2	3	sum
t	6	3	2	4	
Phát bài 0	6				6
Phát bài 1	6	3			9
Phát bài 2	6	3	2		11
Phát bài 3	6	3	2	4	15
Total					41

Cách ghi 2

Bài	2	1	3	0	sum
t	2	3	4	6	
Phát bài 0	2	3	4	6	15
Phát bài 1	2	3			5
Phát bài 2	2				2
Phát bài 3	2	3	4		9
Total					31

```
#include <iostream>
#include <fstream>
// #include <bitset>
// #include <cmath>
// #include <windows.h>
// #include <cstring>
#include <bits/stdc++.h>
// #include <time.h>
// #include <algorithm>

using namespace std;

#define Open() ifstream f("MUSIC.INP")
#define Close() f.close()

int *t, *id;

bool Less(int i, int j) {
    return t[i] < t[j];
}
```

```

void Print(int x[], int d, int c, const char * msg = "") {
    cout << msg;
    for (int i = d; i <= c; ++i) {
        cout << " " << x[i];
    }
}

int Music() {
    Open();
    int n; // so ban nhac
    f >> n;
    cout << "\n n = " << n;
    t = new int[n]; id = new int[n];
    for (int i = 0; i < n; ++i) {
        f >> t[i];
        id[i] = i;
    }

    Print(t, 0, n-1, "\n t: ");
    Print(id, 0, n-1, "\n id: ");
    sort(id, id+n, Less);
    Close();
    Print(id, 0, n-1, "\n Ghi cac ban nhac nhu sau:");
}

main() {
    Music();
    // -----
    cout << endl << "\n\n T h e   E n d \n";
    return 0;
}

```

Result n = 4

```

t:  6 3 2 4
id: 0 1 2 3
Ghi cac ban nhac nhu sau: 2 1 3 0
T h e   E n d

```

Độ phức tạp tính toán. Sắp tăng dãy n phần tử cần n^2 phép so sánh. Nếu dùng các giải thuật nhanh thì cần $n\log(n)$ phép so sánh.

Problem 2. Int To Rome

Chuyển đổi số nguyên dương sang dạng số La Mã.

```

IntToRome(2021) = "MMXXI"
IntToRome(3964) = "MMMCMXLIV"

```

Algorithm Rót nước

Program

```

/*****
Name: INITOROME.CPP
Copyright: (C) 2021

```

```

Author: DevcFan
Date: 11/10/21 10:06
Description:
int to Rome
IntToRome(2021) = "MMXXI"
IntToRome(3964) = "MMMCMXLIV"
*****/

#include <iostream>
// #include <fstream>
// #include <bitset>
// #include <cmath>
// #include <windows.h>
// #include <cstring>
#include <bits/stdc++.h>
// #include <time.h>
// #include <algorithm>

using namespace std;

const int MN = 13;
// Roles 49 AB = B-A; AB = A+B
int ival[] = {1000,900,500,400,100,90, 50, 40, 10, 9, 5, 4,1};
string rval[] = {"M","CM","D","CD","C","XC","L",
"XL","X","IX","V","IV","I"};

void Go() {
    cout << " ? ";
    fflush(stdin);
    if (cin.get() == '.') exit(0);
}

string IntToRome(int b) {
    string r = "";
    for (int i = 0; i < MN; ++i) {
        while (b >= ival[i]) {
            b -= ival[i];
            r += rval[i];
        }
    }
    return r;
}

main() {
    int n = 2021;
    cout << "\n " << n << " -> " << IntToRome(n); // "MMXXI"
    n = 3964;
    cout << "\n " << n << " -> " << IntToRome(3964); // "MMMCMXLIV"
    for (int i = 1; i <= 4000; ++i) {
        cout << "\n " << i << " -> " << IntToRome(i); Go();
    }
    // -----
    cout << endl << "\n\n T h e   E n d \n";
    return 0;
}

```

Result

```
2021 -> MMXXI
3963 -> MMMCMLXIII
1 -> I ?

2 -> II ?

3 -> III ?

4 -> IV ?

5 -> V ?

6 -> VI ?

7 -> VII ?
```

Problem 3. Ba lô

Bài toán ba lô có nhiều phiên bản. Mục này trình bày một phiên bản phù hợp với phương pháp tham.

Có n vật mã số từ 0 đến $n-1$, trọng lượng lần lượt là $w = (w_0, w_0, \dots, w_{n-1})$. Mỗi vật i có giá trị v_i , $0 \leq i < n$. Cho một ba lô có sức mang tối đa m . Cần chọn cả hay một phần của mỗi vật để xếp vào ba lô sao cho tổng giá trị thu được là lớn nhất.

Algorithm Greedy

$x_i = v_i/w_i$ là đơn giá của vật i . Ta ưu tiên cho những vật có đơn giá cao.

Lần lượt xét vật i có đơn giá cao trở xuống

Nếu ba lô còn xếp được thì xếp.

Program

```
#include <iostream>
#include <fstream>
// #include <bitset>
// #include <cmath>
// #include <windows.h>
// #include <cstring>
#include <bits/stdc++.h>
// #include <time.h>
// #include <algorithm>

using namespace std;

#define Open() ifstream f("BALO.INP")
#define Close() f.close()

int n;
double m; // sức mang của balo
double *w; // trọng lượng
double *v; // giá trị
int *id;
```

```

double *x;

void Go() {
    cout << " ? ";
    fflush(stdin);
    if (cin.get() == '.') exit(0);
}

void Print(int x[], int d, int c, const char * msg = "") {
    cout << msg;
    for (int i = d; i <= c; ++i) {
        cout << " " << x[i];
    }
}

void Print(double x[], int d, int c, const char * msg = "") {
    cout << msg;
    for (int i = d; i <= c; ++i) {
        cout << " " << x[i];
    }
}

void Print(double x[], int id[], int d, int c, const char * msg = "") {
    cout << msg;
    for (int i = d; i <= c; ++i) {
        cout << " " << x[id[i]];
    }
}

bool Great(int i, int j) {
    return x[i] > x[j];
}

int Balo() {
    Open();
    f >> n >> m;
    cout << "\n n = " << n << " m = " << m;
    w = new double[n]; // trong luong
    v = new double[n]; // gia tri
    id = new int[n];
    x = new double[n]; // don gia
    for (int i = 0; i < n; ++i) {
        f >> w[i]; // trong luong
        id[i] = i;
    }
    for (int i = 0; i < n; ++i) {
        f >> v[i]; // gia tri
        x[i] = v[i] / w[i]; // don gia
    }
    Close();
    Print(w, 0, n-1, "\n w: ");
    Print(v, 0, n-1, "\n v: ");
    Print(x, 0, n-1, "\n x: ");
    Print(id, 0, n-1, "\n id: ");
    sort(id, id+n, Great); // sap giam theo x
    Print(id, 0, n-1, "\n dec sorted by id: ");
    Print(x, id, 0, n-1, "\n x: ");
    double vsum = 0;
}

```

```

double *r = new double[n]; // result
double rm = m;
memset(r, 0, n*sizeof(double)); // r = all 0
Print(r, 0, n-1, "\n r: ");
int j;
for (int i = 0; i < n; ++i) {
    j = id[i]; // vat j
    if (w[j] > rm) { // balo het cho
        break;
    }
    // w[j] <= m: balo con cho
    rm -= w[j];
    vsum += v[j];
    r[j] = 1;
    cout << "\n Xep vat " << j;
    cout << "\n Tong gia tri " << vsum;
    cout << "\n Trong luong con lai " << rm;
    Go();
}
// cout << "\n break at j = " << j;
Print(r, 0, n-1, "\n r: ");
if (rm > 0 && j < n) { // con cho cho vat j
    r[j] = rm / w[j];
    vsum += v[j]*r[j];
    rm = 0;
    cout << "\n Xep vat " << j;
    cout << "\n Tong gia tri " << vsum;
    cout << "\n Trong luong con lai " << rm;
    Go();
}
}
}

main() {
    Balo();
    // -----
    cout << endl << "\n\n T h e   E n d \n";
    return 0;
}

```

Result

```

n = 3 m = 20
w: 18 15 10
v: 25 24 15
x: 1.38889 1.6 1.5
id: 0 1 2
dec sorted by id: 1 2 0
x: 1.6 1.5 1.38889
r: 0 0 0
Xep vat 1
Tong gia tri 24
Trong luong con lai 5 ?

r: 0 1 0
Xep vat 2
Tong gia tri 31.5
Trong luong con lai 0 ?

```

Problem 4. Lập lịch theo thời hạn

Có n công việc cần thực hiện trên một máy tính, mỗi việc đòi hỏi đúng 1 giờ máy. Với mỗi việc ta biết thời hạn phải nộp kết quả thực hiện sau khi hoàn thành việc đó và tiền thưởng thu được nếu nộp kết quả trước hoặc đúng thời điểm quy định. Chỉ có một máy tính trong tay, hãy lập lịch thực hiện một số công việc trên máy tính sao cho tổng số tiền thưởng thu được là lớn nhất và thời gian hoạt động của máy là nhỏ nhất. Giả thiết rằng máy được khởi động vào đầu ca, thời điểm $t = 0$ và chỉ tắt máy sau khi đã hoàn thành các công việc đã chọn.

Algorithm Greedy

input					vài phương án								
	0	1	2	3	PA	lịch	Giờ thứ						thưởng
d	1	3	5	1			1	2	3	4	5		
v	15	10	100	27									
					1	1→2			1		2	10+100=110	
					2	0→2	0				2	15+100=115	
					3	0→1→2	0		1		2	15+10+100=125	
					4	3→1→2	3		1		2	27+10+100=137	
					5	3→2	3				2	27+100=127	

Ta ưu tiên cho những việc có tiền thưởng cao, do đó ta sắp các việc giảm dần theo tiền thưởng. Với mỗi việc k ta đã biết thời hạn giao nộp việc đó là $d[k]$. Ta xét trục thời gian s . Nếu hạn nộp $d[k]$ trên trục đó đã bận do việc khác thì ta tìm từ thời điểm $d[k]$ trở về trước một thời điểm có thể thực hiện được việc k đó. Nếu tìm được một thời điểm j như vậy, ta đánh dấu bằng mã số của việc đó trên trục thời gian s , $s[j] = k$. Sau khi xếp việc xong, có thể trên trục thời gian còn những thời điểm rỗi, ta dồn các việc đã xếp về phía trước nhằm thu được một lịch làm việc trù mật, tức là không có giờ trống. Với thí dụ đã cho, $N = 4$, thời hạn giao nộp $d = (1, 3, 5, 1)$ và tiền thưởng $v = (15, 10, 100, 27)$ ta tính toán như sau:

- * Khởi trị: trục thời gian với 5 thời điểm ứng với $d_{\max} = 5$ là thời điểm muộn nhất phải nộp kết quả, $d_{\max} = \max \{ \text{thời hạn giao nộp} \}$, $s[1:5] = (0, 0, 0, 0, 0)$.
- * Chọn việc $k = 2$ có tiền thưởng lớn nhất là $v[2] = 100$. Xếp việc 2 với thời hạn nộp $d[2] = 5$ vào lịch s : $s[5] = 2$. Ta thu được lịch $s = (0, 0, 0, 0, 2)$.
- * Chọn tiếp việc 3 có tiền thưởng 27. Xếp việc 3 với thời hạn $d[3] = 1$ vào s : $s[1] = 3$. Ta thu được $s = (3, 0, 0, 0, 2)$.
- * Chọn tiếp việc 0 có tiền thưởng 15 và thời hạn nộp $d[1] = 1$. Ta thấy không thể xếp được việc 0 vì từ thời điểm 1 trở về trước trục thời gian đã kín. Ta thu được s không đổi, $s = (3, 0, 0, 0, 2)$.
- * Chọn nốt việc 1 có tiền thưởng 10. Xếp việc 1 với thời hạn $d[1] = 3$ vào s : $s[3] = 1$, ta thu được $s = (3, 0, 1, 0, 2)$.
- * Dồn việc trên trục thời gian, ta thu được $s = (3, 1, 2, 0, 0)$.
- * Ca làm việc kéo dài đúng $N = 4$ giờ. Tổng tiền thưởng là $v[3] + v[1] + v[2] = 27 + 10 + 100 = 137$.

Program

```
#include <iostream>
#include <fstream>
// #include <bitset>
// #include <cmath>
#include <windows.h>
// #include <cstring>
#include <bits/stdc++.h> // sort, bitset
// #include <time.h>
// #include <algorithm>

using namespace std;

#define Open() ifstream f("JOBS.INP")
#define Close() f.close()

int n;
int *d; // deadline
int *v; // profit
int *id; // index
int *job;
int dmax;
int *s; // lich
int total;

void Go() {
    cout << " ? ";
    fflush(stdin);
    if (cin.get() == '.') exit(0);
}

void Print(int x[], int d, int c, const char * msg = "") {
    cout << msg;
    for (int i = d; i <= c; ++i) {
        cout << " " << x[i];
    }
}

void Print(int x[], int id[], int d, int c, const char * msg = "") {
    cout << msg;
    for (int i = d; i <= c; ++i) {
        cout << " " << x[id[i]];
    }
}

bool Great(int i, int j) {
    return v[i] > v[j];
}

int GetJobs() {
    job = new int[n];
    s = new int[dmax+1];
    for (int i = 0; i <= dmax; ++i) s[i] = -1;
    Print(s, 0, dmax, "\n s:");
}
```



```

int k;
for (int i = 0; i < n; ++i) {
    k = id[i]; // viec k co profit tot
    cout << "\n viec k = " << k << " han nop: " << d[k];
    for (int j = d[k]; j > 0; --j) {
        if (s[j] < 0) {
            s[j] = k;
            Print(s, 1, dmax, "\n Cac viec da chon s: "); // Go();
            break;
        }
    }
}
// Don viec va tinh tien thuong
total = k = 0;
for (int i = 1; i <= dmax; ++i) {
    if (s[i] != -1) {
        s[++k] = s[i];
        total += v[s[i]];
    }
}
return k;
}

int Jobs() {
    Open();
    f >> n;
    cout << "\n n = " << n;
    d = new int[n]; // deadline
    v = new int[n]; // gia tri
    id = new int[n];
    dmax = 0;
    for (int i = 0; i < n; ++i) {
        f >> d[i];
        if (dmax < d[i]) dmax = d[i];
        id[i] = i;
    }
    for (int i = 0; i < n; ++i) {
        f >> v[i]; // gia tri
    }
    Close();
    Print(d, 0, n-1, "\n deadline d: ");
    Print(v, 0, n-1, "\n profit v: ");
    Print(id, 0, n-1, "\n id: ");
    sort(id, id+n, Great);
    Print(id, 0, n-1, "\n sorted id: ");
    int k = GetJobs(); // 4 2 3 1 Thuong = 137`
    Print(s, 1, k, "\n s: ");
    cout << "\n Tong tien thuong: " << total;
}

main() {
    Jobs();
    // -----
    cout << "\n T h e   E n d \n";
    return 0;
}

```

Result

```
n = 4
deadline d:  1 3 5 1
profit v:  15 10 100 27
id:  0 1 2 3
sorted id:  2 3 0 1
s: -1 -1 -1 -1 -1 -1
việc k =  2 hạn nộp: 5
Cac việc đã chọn s:  -1 -1 -1 -1 2
việc k =  3 hạn nộp: 1
Cac việc đã chọn s:  3 -1 -1 -1 2
việc k =  0 hạn nộp: 1
việc k =  1 hạn nộp: 3
Cac việc đã chọn s:  3 -1 1 -1 2
s:  3 1 2
Tổng tiền thưởng: 137
T h e   E n d
```

Độ phức tạp tính toán

$O(n \log(n))$ cho sắp xếp.

Problem 5. Merge Sort

Trộn hai dãy sắp tăng để thu được dãy sắp tăng.

Algorithm

Sơ tay lấy vật nhẹ, tay rồi cầm vật tiếp.

Program

```
#include <iostream>
#include <fstream>
// #include <bitset>
// #include <cmath>
#include <windows.h>
// #include <cstring>
#include <bits/stdc++.h> // sort, bitset
// #include <time.h>
// #include <algorithm>

using namespace std;

const int MN = 100;
int a[MN] = {2, 7, 8, 8, 10, 15, 20};
int b[MN] = {4, 6, 9, 12, 15, 18, 20, 30};
```

```

int c[MN];

void Go() {
    cout << " ? ";
    fflush(stdin);
    if (cin.get() == '.') exit(0);
}

void Print(int x[], int d, int c, const char * msg = "") {
    cout << msg;
    for (int i = d; i <= c; ++i) {
        cout << " " << x[i];
    }
}

int Merge(int a[], int n, int b[], int m, int c[]) {
    int i, j, k;
    i = j = k = 0;
    while(i < n && j < m) {
        c[k++] = (a[i] <= b[j]) ? a[i++] : b[j++];
    }
    while(i < n) {
        c[k++] = a[i++];
    }
    while(j < m) {
        c[k++] = b[j++];
    }
    return n+m;
}

main() {
    int n = 7, m = 8;
    int k = Merge(a, n, b, m, c);
    Print(a, 0, n-1, "\n a: ");
    Print(b, 0, m-1, "\n b: ");
    Print(c, 0, k-1, "\n c: ");
    // -----
    cout << "\n T h e   E n d \n";
    return 0;
}

```

Result

```

a:  2 7 8 8 10 15 20
b:  4 6 9 12 15 18 20 30
c:  2 4 6 7 8 8 9 10 12 15 15 18 20 20 0
T h e   E n d

```

Độ phức tạp tính toán

Thuật toán duyệt mỗi mảng một lần: $O(m+n)$.

Note

Số phép gán trị cho $c = n + m = \#a + \#b = \#c$.

Problem 6. n-Array Merge

Cần trộn n mảng mã số $1..n$ với số phần tử s_1, s_2, \dots, s_n được sắp tăng để thu được mảng C sắp tăng với số phép gán C ít nhất.

	Array 1	Array 2	Array 3	Array 4
size	5	1	2	6

Phương án 1.

✂ $A1 (+) A2 \rightarrow C: 5 + 1 = 6$
✂ $C (+) A3 \rightarrow C: 6 + 3 = 9$
✂ $C (+) A4 \rightarrow C: 9 + 6 = 15$
Sum = $9 + 6 + 15 = 30$

Phương án 2.

✂ $A1 (+) A3 \rightarrow C: 5 + 2 = 7$
✂ $C (+) A4 \rightarrow C: 7 + 6 = 13$
✂ $C (+) A2 \rightarrow C: 13 + 1 = 14$
Sum = $7 + 13 + 14 = 34$

Phương án 3 (tối ưu).

✂ $A2 (+) A3 \rightarrow C: 1 + 2 = 3$
✂ $C (+) A1 \rightarrow C: 3 + 5 = 8$
✂ $C (+) A4 \rightarrow C: 8 + 6 = 14$
Sum = $3 + 8 + 14 = 25$

Algorithm (Huffman)

Khởi tạo dãy s chứa các size gọi là mảng trọng số $s = (5, 1, 2, 6)$

Lặp $n-1$ lần

Chọn 2 phần tử $s[i]$ và $s[j]$ nhỏ nhất trong số các phần tử còn lại
Đánh dấu i và j
Thêm phần tử mới có trọng số $s[i]+s[j]$ vào s

	size	step	
1	5	2	Step 1. Chọn min $i = 2, j = 3$ Thêm phần tử 5, trọng số $1+2 = 3$
2	1	1	$2 (+) 3 \rightarrow 5$
3	2	1	Step 2. Chọn min $i = 1, j = 5$
4	6	3	Thêm phần tử 6, trọng số $5+3 = 8$
5	3	2	$1 (+) 5 \rightarrow 6$
6	8	3	Step 3. Chọn min $i = 4, j = 6$

7	14		Thêm phần tử 7, trọng số $6+8 = 14$ 4 (+) 6 \rightarrow 7 Kết quả 2(+)3(+)4(+),
---	----	--	-----------------------------------------------------------------------------------------

Program

```
#include <iostream>
#include <windows.h>

using namespace std;

const int MN = 100;
int size[MN] = {5, 1, 2, 6};
int n = 4;

void Go() {
    cout << " ? ";
    fflush(stdin);
    if (cin.get() == '.') exit(0);
}

void Print(int x[], int d, int c, const char * msg = "") {
    cout << msg;
    for (int i = d; i <= c; ++i) {
        cout << " " << x[i];
    }
}

void Print(bool x[], int d, int c, const char * msg = "") {
    cout << msg;
    for (int i = d; i <= c; ++i) {
        cout << " " << x[i];
    }
}

int Min(int s[], bool mark[], int k) {
    int val = 99999999;
    int j;
    for (int i = 0; i <= k; ++i) {
        if (!mark[i]) {
            if (s[i] < val) {
                j = i;
                val = s[j];
            }
        }
    }
    return j;
}

int Huffman(int n) {
    int m = n + n - 1;
    int s[m];
    memcpy(s, size, m*sizeof(int)); // copy size -> s
    bool mark[m];
    memset(mark, false, sizeof(mark));
    int i, j, c = 0; // so phep gan
    for (int k = n-1; k < m-1; ++k) {
```

```

        i = Min(s,mark,k);
        mark[i] = true;
        j = Min(s,mark,k);
        mark[j] = true;
        s[k+1] = s[i]+s[j];
        c += s[k+1];
        cout << "\n " << i << " (+) " << j << " -> " << k+1
                << " Chi phi " << c;

        Go();
    }
    cout << "\n Total: " << c;
    return c;
}

main() {
    Huffman(n);
    // -----
    cout << "\n T h e   E n d \n";
    return 0;
}

```

Result

```

1 (+) 2 -> 4 Chi phi 3 ?
4 (+) 0 -> 5 Chi phi 11 ?
3 (+) 5 -> 6 Chi phi 25 ?

Total: 25
T h e   E n d

```

Độ phức tạp tính toán

Thuật toán duyệt $n-1$ lần, mỗi lần tìm min cần n phép so sánh, vậy độ phức tạp là $O(n^2)$.

Problem 7. Cây khung cực tiểu

Cho đơn đồ thị vô hướng, liên thông n đỉnh mã số từ 0 đến $n - 1$, m cạnh mã số từ 0 đến $m - 1$. Đồ thị được gọi là *đơn* nếu giữa hai đỉnh có tối đa một cạnh nối hai đỉnh đó. Đồ thị được gọi là *liên thông* nếu ta cầm một đỉnh bất kỳ rồi nhắc đỉnh đó lên thì toàn bộ đồ thị được nhắc theo.

Cây khung của đồ thị n đỉnh chứa đúng n đỉnh và một số ít nhất các cạnh của đồ thị sao cho vẫn đảm bảo tính liên thông.

Nhận xét 1. Nếu đơn đồ thị liên thông có n đỉnh thì cây khung có đúng $n-1$ cạnh.

Thuật toán xây dựng cây khung (Kruskal)

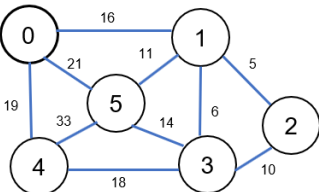
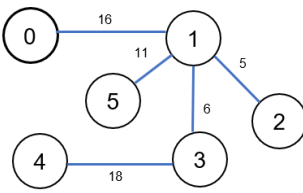
Input: Graph G
Output: Spanning Tree S
begin

Khởi trị cây khung S rỗng.
Lấy từng cạnh (x,y) ghép vào S
sao cho không sinh ra chu trình.
return S
end Spanning

Vận dụng kỹ thuật Find – Union quản lý các tập đỉnh liên thông.

Độ phức tạp tính toán: Duyệt m cạnh, mỗi cạnh kiểm tra n đỉnh: $O(mn)$.

Cây khung cực tiểu: Dùng lại thuật toán Kruskal duyệt các cạnh theo độ dài tăng dần.

GRAPH.INP	GRAPH	MIN SPANNING TREE
6 10 0 1 16 0 4 19 0 5 21 1 2 5 1 3 6 1 5 11 2 3 10 3 4 18 3 5 14 4 5 33		

Program

```

/*****
Name: SPANNINGTREE.CPP
Copyright: (C) 2021
Author: Devc Fan
Date: 11/10/21 15:20
Description:
Minimal Spanning tree
*****/
#include <iostream>
#include <fstream>
#include <algorithm>

using namespace std;

const int NONE = 0;
const int SPAN = 1;

```

```

typedef struct Edge {
    int X;
    int Y;
    int Len;
    int Type;
} Edge;

Edge *c;
int n, m;
int *d;

void Print(Edge e[], int d, int c, const char * msg = "") {
    cout << msg;
    for (int i = d; i <= c; ++i) {
        cout << "\n " << e[i].X << " - " << e[i].Y << " len = " << e[i].Len
            << " Type: " << e[i].Type;
    }
}

void Swap(int & x, int & y) {
    int t = x;
    x = y; y = t;
}

void Read() {
    ifstream f("GRAPH.INP");
    f >> n >> m;
    cout << "\n " << n << " vertexes and " << m << " edges";
    c = new Edge[m];
    int x, y, d, t;
    for (int i = 0; i < m; ++i) {
        f >> x >> y >> d;
        if (x > y) Swap(x,y);
        c[i].X = x; c[i].Y = y;
        c[i].Len = d;
        c[i].Type = NONE;
    }
    f.close();
}

bool Less(Edge a, Edge b) {
    return a.Len < b.Len;
}

int Find(int x) {
    while (d[x] != x) x = d[x];
    return x;
}

int Union(int x, int y) {
    x = Find(x); y = Find(y);
    if (x == y) return 0;
    if (x < y) d[y] = x;
    else d[x] = y;
    return 1;
}

void Spanning() {
    Read();
}

```



```

Print(c, 0, m-1, "\n Read:\n");
sort(c, c+m, Less); // Sap tang theo len
Print(c, 0, m-1, "\n Inc sorted by Len:\n");
d = new int[n];
for (int i = 0; i < n; ++i) d[i] = i;
for (int i = 0; i < m; ++i) {
    if (Union(c[i].X, c[i].Y))
        c[i].Type = SPAN;
}
Print(c, 0, m-1, "\n Result:\n");
int total = 0;
for (int i = 0; i < m; ++i) {
    if (c[i].Type == SPAN) {
        cout << "\n " << c[i].X << " - " << c[i].Y;
        total += c[i].Len;
    }
}
cout << "\n Total: " << total;
}

main() {
    Spanning();
    // -----
    cout << endl << "\n\n T h e    E n d \n";
    return 0;
}

```

Result

6 vertexes and 10 edges
Read:

```

0 - 1 len = 16 Type: 0
0 - 4 len = 19 Type: 0
0 - 5 len = 21 Type: 0
1 - 2 len = 5 Type: 0
1 - 3 len = 6 Type: 0
1 - 5 len = 11 Type: 0
2 - 3 len = 10 Type: 0
3 - 4 len = 18 Type: 0
3 - 5 len = 14 Type: 0
4 - 5 len = 33 Type: 0

```

Inc sorted by Len:

```

1 - 2 len = 5 Type: 0
1 - 3 len = 6 Type: 0
2 - 3 len = 10 Type: 0
1 - 5 len = 11 Type: 0
3 - 5 len = 14 Type: 0
0 - 1 len = 16 Type: 0
3 - 4 len = 18 Type: 0
0 - 4 len = 19 Type: 0
0 - 5 len = 21 Type: 0
4 - 5 len = 33 Type: 0

```

Result:

```

1 - 2 len = 5 Type: 1
1 - 3 len = 6 Type: 1

```

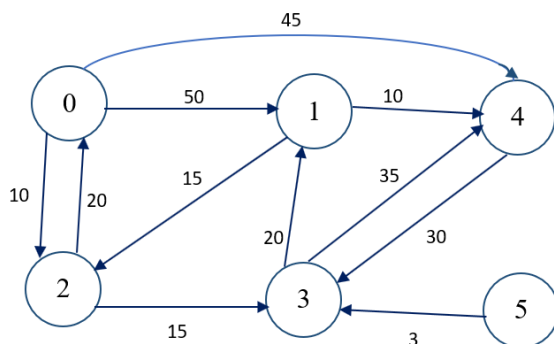
```

2 - 3 len = 10 Type: 0
1 - 5 len = 11 Type: 1
3 - 5 len = 14 Type: 0
0 - 1 len = 16 Type: 1
3 - 4 len = 18 Type: 1
0 - 4 len = 19 Type: 0
0 - 5 len = 21 Type: 0
4 - 5 len = 33 Type: 0
1 - 2
1 - 3
1 - 5
0 - 1
3 - 4
Total: 56
T h e   E n d

```

ProblemAlgorithm Dijkstra

Cho đồ thị có hướng thể hiện một mạng giao thông với các địa điểm (đỉnh) và các đường một chiều thể hiện bằng các cung có hướng $x \rightarrow y$. Trên mỗi cung $x \rightarrow y$ có nhãn thể hiện chi phí đi từ địa điểm x đến địa điểm y . Cần tìm các đường có chi phí ít nhất từ một đỉnh s cho trước đến các đỉnh còn lại.



Dijkstra (sửa đỉnh)						
	(0)	(1)	(2)	(3)	(4)	(5)
c	0	∞	∞	∞	∞	∞
(0)	*	50	10		45	
(2)			*	25		
(3)		45		*		
(1)		*				
(4)					*	
(5)						*

$0 \rightarrow 2$: 10
 $0 \rightarrow 2 \rightarrow 3$: $10 + 15 = 25$
 $0 \rightarrow 2 \rightarrow 3 \rightarrow 1$: $10 + 15 + 20 = 45$
 $0 \rightarrow 4$: $10 + 15 + 10 = 45$
 $0 \rightarrow 5$: ∞

Kí hiệu $C(s, x)$ là chi phí ít nhất từ đỉnh xuất phát s đến đỉnh x . Ta thấy

$C(s, s) = 0$: Từ s đến s không mất chi phí nào.

Muốn đi theo chi phí tối thiểu từ s đến y thì ta chọn chi phí ít nhất từ s đến các đỉnh x sát trước đỉnh y cộng thêm một chi phí cho cung đường $x \rightarrow y$:

$$C(s, x) = \min \{C(y) + d(x, y), y \rightarrow x\}$$

trong đó y là đỉnh sát trước đỉnh x , $d(y, x)$.

Muốn giải trình đường đi: Đặt con trỏ trước $\text{pred}[y] = x$ có nghĩa $x \rightarrow y$.

Program

```

/*****
Name: DIJKSTRA.CPP
Copyright: (C) 2021
Author: Devc Fan
Date: 11/10/21 15:20
Description:
Single Source Shortest Paths
*****/
#include <iostream>
#include <fstream>
#include <algorithm>
#include <windows.h>

using namespace std;

int n; // dinh
int m; // canh

int ** d;
int *c;
int s; // dinh xuất phát
int inf;
bool *taken;
int * pred;

void Go() {
    cout << " ? ";
    fflush(stdin);
    if (cin.get() == '.') exit(0);
}

void Print(int a[], int d, int c, const char * msg = "") {
    cout << msg;
    for (int i = d; i <= c; ++i) {
        cout << " " << a[i];
    }
}

void Print(int *a[], int d, int c, const char * msg = "") {
    cout << msg;
    for (int i = d; i <= c; ++i) {
        Print(a[i], d, c, "\n ");
    }
}

void Read() {
    ifstream f("DIJKSTRA.INP");
    f >> n >> m;
    cout << "\n n = " << n << "    m = " << m;
    d = new int *[n];
    for (int i = 0; i < n; ++i) {
        d[i] = new int[n];
        memset(d[i], 0, n*sizeof(int));
    }
    int x, y, v;

```

```

    inf = 0;
    for (int i = 0; i < m; ++i) {
        f >> x >> y >> v;
        cout << "\n " << x << "->" << y << ": " << v;
        d[x][y] = v;
        inf += v;
    }
    f.close();
}

int GetMin() {
    int vmin = inf;
    int imin;
    for (int i = 0; i < n; ++i) {
        if (!taken[i] && c[i] < vmin) {
            vmin = c[i]; imin = i;
        }
    }
    return imin;
}

void Path(int y) {
    if (pred[y] == y) {
        cout << "\n " << y;
        return;
    }
    Path(pred[y]);
    cout << " -> " << y;
}

void Run(int start) {
    Read();
    Print(d, 0, n-1, "\n Init: ");
    c = new int[n];
    for (int i = 0; i < n; ++i) c[i] = inf;
    taken = new bool[n];
    pred = new int[n];
    for (int i = 0; i < n; ++i) pred[i] = i;
    memset(taken, false, n*sizeof(bool));
    s = start; c[s] = 0;
    int x, val;
    for (int i = 0; i < n; ++i) {
        x = GetMin(); val = c[x];
        taken[x] = true;
        for (int y = 0; y < n; ++y) {
            if (!taken[y] && d[x][y] > 0 && val + d[x][y] < c[y]) {
                c[y] = c[x] + d[x][y];
                pred[y] = x;
            }
        }
    }
    cout << "\n Result:";
    for (int i = 0; i < n; ++i) {
        Path(i);
        if (c[i] < inf) cout << " : " << c[i];
        else cout << " : no path.";
    }
}

```

```

main() {
    Run(0);
    // -----
    cout << endl << "\n T h e   E n d \n";
    return 0;
}

```

Result

```

n = 6    m = 11
0->1: 50
0->2: 10
0->4: 45
1->2: 15
1->4: 10
2->0: 20
2->3: 15
3->1: 20
3->4: 35
4->3: 30
5->3: 3
Init:
 0 50 10 0 45 0
 0 0 15 0 10 0
20 0 0 15 0 0
 0 20 0 0 35 0
 0 0 0 30 0 0
 0 0 0 3 0 0
Result:
0 : 0
0 -> 2 -> 3 -> 1 : 45
0 -> 2 : 10
0 -> 2 -> 3 : 25
0 -> 4 : 45
5 : no path.

T h e   E n d

```