**Minimum Spanning Tree &&**

**第二小的Spanning tree**

#include<cstdio>

#include<cstdlib>

#include<cstring>

#define SE struct Edge

SE

{

//點a到點b距離c

//e在算第二小生成樹會用到

**int** a,b,c,e;

};

SE M[10005];

**int** INF = 1<<30;

**using** namespace std;

**int** n,m;

**int** f[103];

**int** mst[103];

**int** findset(**int** x)

{

**int** a = f[x];

**while**(a!=f[a])

a = findset(f[a]);

f[x] = a;

**return** a;

}

**int** uniset(**int** x,**int** y)

{

f[x] = y;

}

**int** MST(**void**)

{

**int** i,j,k;

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

{

f[i] = i;

}

**int** a,b;

k = 0;

i = j = 0;

**while**(i<n-1&&j<m)

{

**if**(M[j].e==0)

{

a = findset(M[j].a);

b = findset(M[j].b);

**if**(a!=b)

{

k+=M[j].c;

mst[i++] = j;

uniset(a,b);

}

}

j++;

}

**if**(i<n-1)

**return** INF;

**return** k;

}

**int** second\_MST(**void**)

{

**int** B[103];

**int** i,j,k;

**for**(i=0;i<n-1;i++)

B[i] = mst[i];

**int** min = INF;

**for**(i=0;i<n-1;i++)

{

M[B[i]].e = 1;

**int** minf = MST();

**if**(minf<min)

min = minf;

M[B[i]].e = 0;

}

**return** min;

}

**int** cmp(**const** **void**\*a,**const** **void** \*b)

{

SE x = \*((SE\*)a);

SE y = \*((SE\*)b);

**return** x.c-y.c;

}

**int** main(**void**)

{

**int** t,i,j,k;

**int** a,b,c;

**int** ans,ans1;

scanf("%d",&t);

**while**(t--)

{

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

**for**(i=0;i<m;i++)

{

scanf("%d %d %d",&a,&b,&c);

M[i].a= a;

M[i].b = b;

M[i].c = c;

M[i].e = 0;

}

qsort(M,m,**sizeof**(M[0]),cmp);

ans = MST();

ans1 = second\_MST();

printf("%d %d\n",ans,ans1);

}

**return** 0;

}

**硬幣有幾種付法**

#include<iostream>

#include<limits.h>

#include<string.h>

**using** **namespace** std;

**int** dp[1000];

**int** main()

{

**int** c[5]={1,5,10,20,50};

memset(dp,0,**sizeof**(dp));

dp[0]=1;

**int** n=5;

**for**(**int** i=0;i<n;i++){

**for**(**int** j=c[i];j<=50;j++){

dp[j]+=dp[j-c[i]];}}

**for**(**int** i=0;i<=50;i++)

cout<<dp[i]<<endl;

**return** 0;}

**0-1KNAPACK**

#include<iostream>

#include<cstdio>

#include<string.h>

#include<math.h>

**using** **namespace** std;

**int** max(**int** x, **int** y)

{ **if**(x > y) **return** x;

**else** **return** y;}

**const** **int** N = 100, W = 100;

**int** cost[N], weight[N];

**int** c[N + 1][W + 1];

//c[n][w] 取前n個物品 可耐重w時最大的利潤

**void** knapsack(**int** n, **int** w){

memset(c, 0, **sizeof**(c));

**for** (**int** i = 0; i < n; ++i)

**for** (**int** j = 0; j <= w; ++j)

**if** (j - weight[i] < 0)

c[i+1][j] = c[i][j];

**else**

c[i+1][j] =

max(c[i][j],c[i][j - weight[i]] + cost[i]);

cout << c[n][w]<<endl;

}

**int** main(){

**int** n;

**while**(cin>>n){

**for** (**int** i = 0; i < n; ++i){

cin>>weight[i];

cin>>cost[i];}

knapsack(n,100);}

**return** 0;}

**SIEVE OF PRIME**

#include <bits/stdc++.h>

**using** **namespace** std;

**void** SieveOfEratosthenes(**int** n){

**bool** prime[n+1];

memset(prime, **true**, **sizeof**(prime));

**for**(**int** p=2; p\*p<=n; p++){

**if** (prime[p] == **true**){

**for**(**int** i=p\*2;i<=n;i+= p) prime[i] = **false**;

}}

// Print all prime numbers

**for** (**int** p=2; p<=n; p++)

**if** (prime[p]) cout << p << " ";}

**INPUT不定空白**

#include<iostream>

#include<cstdio>

#include<cstring>

#include<sstream>

**using** **namespace** std;

**int** main(){

string line;

**while**(getline( cin, line) )

//利用 getline讀入一行

{

istringstream in( line );

**int** sum = 0, x;

**while** (in >> x) sum += x;

cout<< sum << endl;}}

**PRIORITY QUEUE**

#include<queue>

#include<algorithm>

#include<iostream>

**using** **namespace** std;

**struct** cmp1{

**bool** **operator**()(**int** a,**int** b)

{**return** a<b;}};

**int** main(){

priority\_queue <**int**,vector<**int**>,cmp1> max\_pq;

**for**(**int** i=0;i<10;i++)

max\_pq.push(i);

max\_pq.pop();

cout<<max\_pq.top();}

**INPUT不定逗號區別版**

**int** a[50];

**char** c=' ';

**int** main(){

**int** counter = 0;

stringstream ss;

string s;

memset(a,0,**sizeof**(a));

cin >> s;

ss << s;

**while**(ss>> a[counter++]){

ss >>c;};

**for**(**int** i=0;i<counter-1;i++)

cout << a[i] << endl;

scanf("[^ ]",a[0]);}

**BIGIDECIMAL**

package bigdc;

**import** java.io.\*;

**import** java.math.\*;

**public** **class** bigdc {

**public** **static** **void** main(String[] args){

//BigDecimal.setScale();

BigDecimal a=**new** BigDecimal("0.111111111");

BigDecimal b=**new** BigDecimal("1.111111111111");

b=b.add(a);

System.out.println(b);

b=b.setScale(10,BigDecimal.ROUND\_DOWN);

//b.scale();

System.out.println(b);}}

ROUND\_CEILING 無條件進位

(正數<-零<-負數[負數趨向正數])

ROUND\_FLOOR 無條件捨去

(正數->零->負數[負數趨向負到一個不行])

ROUND\_UP 無條件進位(正數<-零->負數)

ROUND\_DOWN 無條件捨去(正數->零<-負數) ROUND\_HALF\_UP 四捨五入(正數<-零->負數) ROUND\_HALF\_EVEN 四捨五入(以2.5為例，會取成2) ROUND\_HALF\_DOWN 四捨五入(正數->零<-負數) ROUND\_UNNECESSARY 不需要round的功能

**LIS**

#include <stdio.h>

#include <memory.h>

#define MAX\_N 1000

**int** b[MAX\_N + 10];

**int** aMaxLen[MAX\_N + 10];

**int** main() {

**int** N;

scanf("%d", & N);

**for**( **int** i = 1;i <= N;i ++ )

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

aMaxLen[1] = 1;

**for**( **int** i = 2; i <= N; i ++ ) {

**int** nTmp = 0;

**for**( **int** j = 1; j < i; j ++ ) {

**if**( b[i] > b[j] ) {

**if**( nTmp < aMaxLen[j] )

nTmp = aMaxLen[j]; } }

aMaxLen[i] = nTmp + 1; }

**int** nMax = -1;

**for**(**int** i = 1;i <= N;i ++ )

**if**( nMax < aMaxLen[i])

nMax = aMaxLen[i];

printf("%d\n", nMax); }

INPUT 7 1 7 3 5 9 4 8

OUTPUT 4

**N-QUEEN**

#include <iostream>

#include <cstdio>

#include <cstdlib>

#include <cstring>

**using** **namespace** std;

**long** sum = 0, upperlim = 1;

**void** test(**long** row, **long** ld, **long** rd){

**if** (row != upperlim){

**long** pos = upperlim & ~(row | ld | rd);

**while** (pos){

**long** p = pos & -pos;

pos -= p;

test(row+p,(ld +p) << 1, (rd + p) >> 1);}}

**else** sum++; }

**int** main(**int** argc, **char** \*argv[]){

**int** n = 15;

**if** (argc != 1)

n = atoi(argv[1]);

**if** ((n < 1)||(n > 32)){

printf(" ONLY1-32\n");

exit(-1);}

printf("%dQUEEN\n", n);

upperlim = (upperlim << n) - 1;

test(0, 0, 0);

printf("%ld",sum);}

**字串匹配**

**int** failure[100];

**void** Morris\_Pratt(string& t, string& p){

**if** (p.size() > t.size()) **return**;

// 預先計算 P 的 failure function，O(P)。

**for** (**int** i=1, j=failure[0]=-1; i<p.size(); ++i){

**while** (j >= 0 && p[j+1] != p[i])

j = failure[j];

**if** (p[j+1] == p[i]) j++;

failure[i] = j;}

// 進行字串匹配，O(T)。

**for** (**int** i=0, j=-1; i<t.size(); ++i)

// for (int i=0, j=-1; i-j-1+p.size()<=t.size(); ++i)

{

// 先試 p[0...j] 的「最長的相同前綴後綴」，

// 再試「次長的相同前綴後綴」，

// 再試「次次長的相同前綴後綴」……

// 直到試成功為止。

**while** (j >= 0 && p[j+1] != t[i])

j = failure[j];

// t[i] 終於有用處了，終於可以加長！

**if** (p[j+1] == t[i]) j++;

// 匹配到P！

**if** (j == p.size()-1){

cout << "P出現位置" << i - p.size() + 1;

// 如果字串結尾不是'\0'的時候，就必須挪動 P。

// 如果字串結尾是'\0'的時候，就能省略這一行。

j = failure[j];}}}

**最長子字串(印數量)**

#include <stdio.h>

#include <string.h>

#define MAX\_LEN 1000

**char** sz1[MAX\_LEN];

**char** sz2[MAX\_LEN];

**int** aMaxLen[MAX\_LEN][MAX\_LEN];

main() {

**while**( scanf("%s%s", sz1+1 ,sz2+1 ) > 0 ) {

**int** nLength1 = strlen( sz1+1);

**int** nLength2 = strlen( sz2+1);

**int** nTmp;

**int** i, j;

**for**( i = 0;i <= nLength1; i ++ )

aMaxLen[i][0] = 0;

**for**( j = 0;j <= nLength2; j ++ )

aMaxLen[0][j] = 0;

**for**( i = 1;i <= nLength1;i ++ ) {

**for**( j = 1; j <= nLength2; j ++ ) {

**if**( sz1[i] == sz2[j] )

aMaxLen[i][j] = aMaxLen[i-1][j-1] + 1;

**else** {

**int** nLen1 = aMaxLen[i][j-1];

**int** nLen2 = aMaxLen[i-1][j];

**if**( nLen1 > nLen2 )

aMaxLen[i][j] = nLen1;

**else**

aMaxLen[i][j] = nLen2; }}}

printf("%d\n", aMaxLen[nLength1][nLength2]); } }

INPUT abcfbc abfcab

programming contest

abcd mnp

OUTPUT 4 2 0

**m-coloring(backtraking)**

#include<iostream>

#include<string.h>

**using** **namespace** std;

**bool** w[300][300]= {};

**int** vcolor[90000]={};

**int** checkedge=0;

**int** checkedgesum=0;

**int** ans=0;

**bool** promising(**int** i){

**int** j;

**bool** switchs;

switchs=**true**;

j=0;

**while**(j<i &&switchs){

**if**(w[i][j]&&vcolor[i]==vcolor[j])

switchs=**false**;

j++;}

**return** switchs;}

**int** m\_coloring(**int** i,**int** n){

**int** color;

**if**(promising(i))

**if**(i==n){

/\*for(int a=1;a<=n;a++)

         cout<<vcolor[a]<<" ";

         cout<<endl;\*/

**for**(**int** a=0;a<n;a++){

**for**(**int** b=0;b<n;b++){

**if**(w[a][b]==**true**) {

**if**(vcolor[a+1]==vcolor[b+1]){

checkedge=1;}}}}

**if**(checkedge==1){

checkedgesum++;

checkedge=0; }

ans++;}

**else**{

**for**(color=1;color<=2;color++){

//color2兩種顏色

vcolor[i+1]=color;

m\_coloring(i+1,n);}}}

**常用FUNCTION**

字符測試篇

Isalnum　isalpha　isascii　iscntrl　isdigit　isgraphis　islower　isprint　isspace

Ispunct　isupper　isxdigit

字符串轉換篇

Atof　atoi　atoll　gcvt　strtod　strtol　strtoul　toascii　tolower　toupper

內存控制篇

Calloc　free　getpagesize　malloc　mmap　munmap

內存及字符串操作篇

Bcmp　bcopy　bzero　index　memccpy　memchr　memcmp　memcpy　memmove

Memset　rindex　strcasecmp　strcat　strchr　strcmp　strcoll　strcpy　strcspn　strdup

Strlen　strncasecmp　strncat　strncpy　strpbrk　strrchr　strspn　strstr　strtok

常用數學函數篇

Abs　acos　asin　atan　atan2　ceil　cos　cosh　exp　frexp　ldexp　log　log10

Pow　sin　sinh　sqrt　tan　tanh

**int** main(){

**int** edge,node,x,y;

**while**(cin>>node){

memset(w,0,**sizeof**(w));

checkedgesum=0;

ans=0;

**if**(node==0){**break**;}

**else**{

cin>>edge;

**for**(**int** i=0; i<edge; i++){

//無向圖

cin>>x>>y;

w[x][y]=**true**;

w[y][x]=**true**;}}

m\_coloring(0,node);

**if**(checkedgesum==ans)

cout<<"NOT BICOLORABLE."<<endl;

**else** cout<<"BICOLORABLE."<<endl;}}

**SUM OF SUBSET**

**int** main(){

**int** weights[] = {15, 22, 14, 26, 32, 9, 16, 8};

**int** target = 53;

**int** size = ARRAYSIZE(weights);

generateSubsets(weights, size, target);

printf("Nodes generated %d\n", total\_nodes);

**return** 0;}

#include <stdio.h>

#include <stdlib.h>

#define ARRAYSIZE(a) (sizeof(a))/(sizeof(a[0]))

**static** **int** total\_nodes;

// prints subset found

**void** printSubset(**int** A[], **int** size){

**for**(**int** i = 0; i < size; i++){

printf("%\*d", 5, A[i]);}

printf("\n");}

**int** comparator(**const** **void** \*pLhs, **const** **void** \*pRhs){

**int** \*lhs = (**int** \*)pLhs;

**int** \*rhs = (**int** \*)pRhs;

**return** \*lhs > \*rhs;}

**void** subset\_sum(**int** s[], **int** t[],

**int** s\_size, **int** t\_size,

**int** sum, **int** ite,

**int** **const** target\_sum)

{

total\_nodes++;

**if**( target\_sum == sum ){

printSubset(t, t\_size);

**if**( ite + 1 < s\_size && sum - s[ite] + s[ite+1] <= target\_sum ){

subset\_sum(s, t, s\_size, t\_size-1, sum - s[ite], ite + 1, target\_sum);}

**return**;}

**else**{

**if**( ite < s\_size && sum + s[ite] <= target\_sum ){

**for**( **int** i = ite; i < s\_size; i++ ){

t[t\_size] = s[i];

**if**( sum + s[i] <= target\_sum ){

subset\_sum(s, t, s\_size, t\_size + 1, sum + s[i], i + 1, target\_sum);}}}}

}

**void** generateSubsets(**int** s[], **int** size, **int** target\_sum)

{

**int** \*tuplet\_vector = (**int** \*)malloc(size \* **sizeof**(**int**));

**int** total = 0;

qsort(s, size, **sizeof**(**int**), &comparator);

**for**( **int** i = 0; i < size; i++ ){

total += s[i];}

**if**( s[0] <= target\_sum && total >= target\_sum ){

subset\_sum(s, tuplet\_vector, size, 0, 0, 0, target\_sum);}

free(tuplet\_vector);}

**strtok**

**Largest Empty Rectangle**

**bool** array[10+2][10+2];

**int** width[10+2][10+2];

**int** largest\_empty\_rectangle(){

/\* 計算所有橫條當中，每個位置往左可延伸的長度。 \*/

**for** (**int** i=1; i<=10; ++i)

**for** (**int** j=1; j<=10; ++j)

**if** (array[i][j])

width[i][j] = width[i][j-1] + 1;

**else**

width[i][j] = 0;

/\* 計算每個位置當作矩形右下角頂點時的最大矩形面積。 \*/

// 最大矩形面積，初始化為最小值

**int** area = 0;

// 窮舉矩形右下角頂點的位置

**for** (**int** i=1; i<=10; ++i)

**for** (**int** j=1; j<=10; ++j){

**int** w = 1e9;

**for** (**int** h=1; i-h+1 >= 0; ++h) {

**if** (width[i-h+1][j] == 0) **break**;

w = min(w, width[i-h+1][j]);

area = max(area, w\*h); }}

**return** area;}

呼叫形式：strtok(str,delim)

功　　能：將字串str以delim分割開

#include "string.h"

#include "stdlib.h"

#include "stdio.h"

**int** main()

{

**char** str[]="00:22:33:4B:55:5A";

**char** \*delim = ":";

**char** \* pch;

 printf ("Splitting string \"%s\" into tokens:\n",str);

 pch = strtok(str,delim);

**while** (pch != NULL)

 {

   printf ("%s\n",pch);

   pch = strtok (NULL, delim);

/\*當第一個參數為NULL時，代表指向上一次分割後剩下的字串\*/

 }

 system("pause");

**return** 0;

}

**fgets**

呼叫形式：fgets(buf, sizeof(buf), p\_file)

功　　能：將p\_file所指的檔案，讀buf個大小的內容到buf裡面，但碰到\n便停止，因此每次只

　　　　　會讀一行

注　　意：buf的大小要比一行內容的大小還大

**while** (fgets(buf, **sizeof**(buf), p\_file) != NULL)

**fseek**

呼叫形式：fseek(p\_file, 0L, SEEK\_SET);

功　　能：將檔案指標p\_file重新設定在開頭，0L代表距離開頭0的位置

注　　意：第三個參數有三種，各有不同意義

**memcmp**

呼叫形式：memcmp(mod->grid[i].id,mod->grid[j].id,sizeof(char)\*10)

功　　能：比較mod->grid[i]跟mod->grid[j].id的大小，從開頭開始比sizeof(char)\*10的大小

**memcpy**

呼叫形式：memcpy(temp ,&mod->grid[i],sizeof(struct grid)/sizeof(char));

功　　能：將&mod->grid[i]這個位置的內容複製到temp這個位置

　共複製sizeof(struct grid)/sizeof(char)個字節

**C++**

**str**

呼叫形式：stream.str()

函式庫名：<sstream>

功　　能：將字串流轉為字串

注　　意：可以丟入很多不同type的東西後，再一次轉成一個字串

stringstream stream;

stream **<<** price;

**return** stream.str();

**strpbrk**（查找字符串中第一個出現的指定字符）

相關函數

index，memchr，rindex，strpbrk，strsep，strspn，strstr，strtok

表頭文件

#include <include.h>

定義函數

char \*strpbrk(const char \*s,const char \*accept);

函數說明

strpbrk()用來找出參數s 字符串中最先出現存在參數accept 字符串中的任意字符。

返回值

如果找到指定的字符則返回該字符所在地址，否則返回0。

範例

#include <string.h>

main()

{

char \*s="0123456789012345678901234567890";

char \*p;

p=strpbrk(s,"a1 839"); /\*1會最先在s字符串中找到\*/

printf("%s\n",p);

p=strprk(s,"4398");/\*3 會最先在s 字符串中找到\*/

printf("%s\n",p);

執行

1.23E+29

**strrchr**（查找字符串中最後出現的指定字符）

相關函數

index，memchr，rindex，strpbrk，strsep，strspn，strstr，strtok

表頭文件

#include<string.h>

定義函數

char \* strrchr(const char \*s, int c);

函數說明

strrchr()用來找出參數s字符串中最後一個出現的參數c地址，然後將該字符出現的地址返回。

返回值

如果找到指定的字符則返回該字符所在地址，否則返回0。

範例

#include<string.h>

main()

{

char \*s="0123456789012345678901234567890";

char \*p;

p=strrchr(s,'5');

printf("%s\n",p);

}

執行

567890

**strstr**（在一字符串中查找指定的字符串）

相關函數

index，memchr，rindex，strchr，strpbrk，strsep，strspn，strtok

表頭文件

#include<string.h>

定義函數

char \*strstr(const char \*haystack,const char \*needle);

函數說明

strstr()會從字符串haystack 中搜尋字符串needle，並將第一次出現的地址返回。

返回值

返回指定字符串第一次出現的地址，否則返回0。

範例

#include<string.h>

main()

{

char \* s="012345678901234567890123456789";

char \*p;

p= strstr(s,"901");

printf("%s\n",p);

}

執行

9.01E+21

**FLOW**

int BFS(int s, int t) // 源點與匯點

{

memset(visit, false, sizeof(visit));

queue<int> Q;

visit[s] = true;

path[s] = s;

bottleneck[s] = 1e9;

Q.push(s);

while (!Q.empty())

{

int i = Q.front(); Q.pop();

for (int j=0; j<10; ++j)

// 在剩餘容量的圖上找augmenting path

if (!visit[j] && R[i][j] > 0)

{

visit[j] = true;

path[j] = i;

// 一邊找最短路徑，一邊計算流量瓶頸。

flow[j] = min(flow[i], R[i][j]);

Q.push(j);

if (j == t) return flow[t];

}

}

return 0;

// 找不到augmenting path了，流量為零。

}

int Edmonds\_Karp(int s, int t)

{

memset(F, 0, sizeof(F));

memcpy(R, C, sizeof(C));

int f, df; // 最大流的流量、擴充路徑的流量

for (f=0; df=BFS(s, t); f+=df)

// 更新擴充路徑上每一條邊的流量

for (int i=path[t], j=t; i!=j; i=path[j=i])

{

F[i][j] = F[i][j] + df;

F[j][i] = -F[i][j];

R[i][j] = C[i][j] - F[i][j];

R[j][i] = C[j][i] - F[j][i];

}

return f;

}