ASCII Table : $man ascii

2^32-1=2147483647

printf("%03d", a);空格補 0

printf (“%.2lf”,)小數點後 2 位

struct NAME{ }VARNAME;

typedef struct NAME{ }; NAME VARNAME;

O(N) = 1000000 ~ 2000000 equals to Run Time = 1s

#include <algorithm>

Bubble sort O(N^2)

```
void bubble_sort(int in[], int n)
{
int k, j;
for(k=n-1; k>=1;k--)
      for(j=0; j<k; j++)
            if(in[j]>in[j+1])
                  swap(in[j], in[j+1]);
}
```

Insertion SortO(N^2)

```
void insertion_sort( int n, int a[])
{
int k, j, item;
for ( k = 1; k<n; k++)
{
item = a[k];
      for ( j = k-1; j>=0; j--)
            if( a[j] > item )
                  { a[j+1]=a[j]; }
            else break; // a[j] <= item
                  a[j+1] = item;
}
}
```

Merge Sort O(nlog n)

```
Divide: void mergeSort(int a[], int low, int high) {
int k;
if ( high > low)
{     mergeSort(a, low, (low+high)/2);
      mergeSort(a, 1+(low+high)/2, high);
      merge(a, low, high); }
}
Merge:
void merge (int a[], int low, int high) {
int b[MAXSIZE], i, j, k = -1, mid=(low+high)>>1;
 for (i=low, j=mid+1; i<=mid || j<=high; )
{
    if ( i>(low+high)/2 ) b[++k]=a[j++];
    else if (j>high) b[++k]=a[i++];
    else if (a[i]>=a[j]) b[++k]=a[j++];
    else b[++k]=a[i++];
}
k=0;
for(i=low; i<=high; i++) a[i]=b[k++];
}
```

Quick Sort

```
void q(int a[],int l,int r){
      int p,i,j;
      if(l<r)      {
            i=l;    j=r+1;      p=a[l];
            do {
                  do i++; while(a[i]<p);
                  do j--; while(a[j]>p);
                  if(i<j) swap(a[i],a[j]);
            }while(i<j);
            swap(a[l],a[j]);

            q(a,l,j-1);
            q(a,j+1,r);

      }
}
```

陣列排序 sort(a,a+5)

Stucture排序

#include <algorithm> <- 記得!!!

```
struct NODE
{
int x, y ;
bool operator<(const NODE& t ) const
{
return x < t . x ; (依據 x 由小到大排序)
} return x == p.x ? y < p.y : x < p.x;
}p[MAXN];
sort( p , p + n ) ;
```

stack (queue)

push pop top(front) empty

map#include<map>

map<int, int>QQ; (int 預設 0 string \0)

QQ[1234567] = 123456; QQ.clear()

Vector 排序 push_back erase(v.begin()+i)

```
sort(a.begin(), a.end());
Sort by decreasing order…
bool cmp (int a, int b)
{ return a > b; }
sort(a.begin(), a.end(), cmp);
-----------------------
Sort //by lexical order of name
bool cmp(STUDENT a, STUDENT b)
  { return strcmp(a.name, b.name)<0; }
sort(st.begin(), st.end(), cmp);
struct STUDENT {
  char name[10];
int score;
bool operator<(const STUDENT& t)const
  { return strcmp(name, t.name)<0; }
};
sort(st.begin(), st.end());
```

string

```
strcat(A,B)B 接在 A 後面  strcmp(A, B)比較  0 一樣
char* pch;   pch = strtok(str,字); pch = strtok (NULL, 字);
strcpy(A,B) B 覆蓋 A A=B
b=string(in) char 轉字串  char*c="" s.assign(c);
c=s.c_str()字串轉 char
```

Set

```
#include<set>
//declaration
set<int>my;
my.insert(1); my.insert(5); my.insert(3);
//IO
set<info>::iterator itr;
for(itr=my.begin(); itr!=my.end(); itr++)
cout<<*itr<<" "<<endl; //1, 3, 5
//clear
my.clear();
```

disjoint set

```
void MakeSet(int x)
{   p[x] = x;     rank[x] = 0; }
void Union(int x,int y)
{   Link(FindSet(x),FindSet(y));
int FindSet(int x)
{   if(x!=p[x])
p[x] = FindSet(p[x]);
return p[x];
void Link(int x,int y)
{
if(rank[x]>rank[y])
    p[y] = x;
else
{
    p[x] = y;
    if(rank[x]==rank[y])
        rank[y]++;
}  }
```

Binary search tree

```
 bool canbe(int in)        return in>33;
int binary_search(int beg,int end)
{
      int mid;
      do   {
           //find min->***mid=(beg+end)>>1;
           //find max->***mid=(beg+end+1)>>1;
           if(canbe(mid)) //find min->end=mid;
                          //find max->beg=mid
           else
                //find min->beg=mid+1;***
                //find max->end=mid-1;***
      }while(beg<end)
      return end;
}
```

DFS

```
void dfs(int now)
{   visited[now] = true;
   for(int i=0; i<(int)adj[now].size(); i++)
   {nn int next = adj[now][i];
      if(!visited[next]) dfs(next);
   } }
```

BFS

```
 int visited[n], i;
 main()
   { int v;
     for (i=0; i<n; i++) visited[i] = 0;
     visited[0] = 1;
     Add_Queue(0);
     while (!Queue.empty())
     { v = Queue.front(), Queue.pop();   印出v;
     for (所有與v相鄰的頂點w)
11 { if (visited(w) == 0)
```

質數表

```
#define MAX 1000000
#include<cstring>
bool isprime[MAX];
void Sieve(){ // Time complexity may be (n x sqrt(n))
memset( isprime, true, sizeof(isprime) );
isprime[0] = false;
isprime[1] = false;
for( int i=2; i<MAX; i++)
if( isprime[i] )
for(int j = i + i; j<MAX; j += i)
isprime[j] = false;
}
for( int i=2; i<=sqrt(MAX); i++) for( int j=i * i; j<MAX; j
+= i )
```

Backtracking

```
int solution[MAX]; // a candidate
bool used[MAX]; // constraint
void permutation(int k, int n) //the kth dimension
{
   if (k == n) // it's a solution {
       for (int i=0; i<n; i++)
       cout << solution[i] << " "; cout << endl; }
   else {
       for (int i=0; i<n; i++)
           if (!used[i]){
               used[i] = true; // set constraint
                 solution[k] = i; // set solution
               permutation(k+1, n); // recursive
               used[i] = false; // back up the
constraint }
      }
}
(       solution[n] = true; backtrack(n+1);
      solution[n] = false; backtrack(n+1);      )
```

LIS

```
for(int i=0;i<n;++i){ scanf("%d",&seq);
      if(lis.empty())                lis.push_back(seq);
      else{  f(lis.back()<seq)    lis.push_back(seq);
      else if(lis.back()>seq)
      *lower_bound(lis.begin(),lis.end(),seq)=seq;

      //比n大一點的位置->by binary search
for(int i=0;i<n;++i)  {
            max=0; tmp=-1;
            for(int j=i-1;j>=0;--j)
            {     if(j>=0) //小心記憶體區段錯誤
                  {     if(seq[j]<seq[i])
                        {     if(dp[j]>=max)
                              {           max=dp[j];
                                          tmp=j;
            }     }     }     }
            dp[i]=max+1;
            pi[i]=tmp;      }
```

LCS

```
for(int i=0;i<s1.size();++i)
            {     for(int j=0;j<s2.size();++j)
                  {     if(s1[i]==s2[j])
                              lcs[i+1][j+1]=lcs[i][j]+1;
                        else if(s1[i]!=s2[j])
                        {
      lcs[i+1][j+1]=max(lcs[i][j+1],lcs[i+1][j]);        }
                        if(lcs[i+1][j+1]>ans)
                              ans=lcs[i+1][j+1];
            }     }
```

0/1 Knapsack problem

Top-down

```
int dp[N+1][W+1], v[N], w[N];
// top-down, N items with maximum total weight W
  bool isfind[N][W];
  int knapsack(int n, int m){
     if (m < 0) return -INF; // basic constrain
     if (n == 0 ) return 0;
     if (isfind[n][m]) return dp[n][m] ; // isfind before
dp[n][m]=max(knapsack(n-1,m),knapsack(n-1,m-w[n])+v[n])
// recursively call
     isfind[n][m ] = true; // record isfin
     return dp[n][m]; }
```

Botton-up

```
int c[W+1], v[N], w[N]; // bottom-up
int knapsack( int n, int w){
memset(c, 0, sizeof(c)); // Initialize basic constrain
for (int i = 0; i < n; i++)
for (int j = W; j - w[i] >= 0; j--) // Back to the front
c[j] = max( c[j], c[j - w[i]] + v[i] ); // Update lookup table
return c[w];
```

Priority queue

Priority_queue<NODE> pq;

Pq.push(VAR); pq.empty(); pq.top(); pq.pop();

Minimum Spanning Tree

Kruskal's algorithm

–Sort edge –Check cycle (disjoint set)

```
for(int j=0;j<n;++j){
        makeset(j);
                for(int k=0;k<n;++k){
                        scanf("%d",&r[j][k]);
                        if(r[j][k]!=0 && j>k) {
                e[m].v1=j;e[m].v2=k;e[m].l=r[j][k];m++;}
}               }
                sort(e,e+m);
                for(int j=0;j<m;++j) {
                        if(findset(e[j].v1)==findset(e[j].v2))
                                continue;
                        else {
                             link(findset(e[j].v1),findset(e[j].v2));
                        if(e[j].l>max)
                                        max=e[j].l;
                }       }
```

Prim's algorithm

–1. 所有節點設為未拜訪過

–2. 令d[i] 為到節點i的目前距離，起使皆設為INF

–3. 每次都去找未拜訪過的節點 i ，而且d[i] 最小

–4. 找完後要更新未拜訪過的節點距離 d[j] . if 找到的節點 i 到節點 j 的距離小於d[j] 則要更新d[j]

```
1. void prim()
2. {
3.    for (int i=0; i<9; i++) visit[i] = false;
4.    for (int i=0; i<9; i++) d[i] = 1e9;
5.
6.    d[0] = 0;  // 可以選定任何點作為樹根，這裡以第零點作為樹根。
7.
8.
9.    for (int i=0; i<9; i++)
10.   {
11.      int a = -1, b = -1, min = 1e9;
12.      for (int j=0; j<9; j++)
13.         if (!visit[j] && d[j] < min)
14.         {
15.            a = j;  // 記錄這一條邊
16.            min = d[j];
17.         }
18.
19.      if (a == -1) break; // 與起點相連通的 MST 都已找完
20.      visit[a] = true;
21. //    d[a] = 0;        // 註解後，得到 MST 每條邊權重。
22.
23.      for (b=0; b<9; b++)
24.         // 以下與 Dijkstra's Algorithm 略有不同
25.         if  (!visit[b] && w[a][b] < d[b])
26.         {
27.            d[b] = w[a][b]; // 離樹最近，不是離根最近。
28.            parent[b] = a;
29.         }
30.   }
31. }
```

```
ool SPFA(int start)

    queue<int>v;
    int count[MAXN],nowv,nextv;
    bool inqueue[MAXN];

    memset(count,0,sizeof(count));
    memset(inqueue,0,sizeof(inqueue));
    for(int i=0;i<MAXN;++i)
        dis[i]=INF;
    v.push(start);
    inqueue[start]=true;
    count[start]=1;
    dis[start]=0;

    while(!v.empty())
    {
        nowv=v.front();
        v.pop();
        inqueue[nowv]=false;
        for(int i=0;i<(int)edge[nowv].size();++i)
        {
            nextv=edge[nowv][i].t;
            if(dis[nowv]+edge[nowv][i].w<dis[nextv])
            {
                dis[nextv]=dis[nowv]+edge[nowv][i].w;
                if(!inqueue[nextv])
                {
                    v.push(nextv);
                    inqueue[nextv]=true;
                    ++count[nextv];
                }
                if(count[nextv]>=N)
                    return true;
            }
        }
    }
    return false;
```

```
int Dijkstra(int start){
    int nowv,nowd,nextv;
    priority_queue<NODE>pq;
    for(int i = 0; i < MAXN; ++i)
        dis[i]=INF;
    TMP.id=start,TMP.d=0;
    pq.push(TMP);
    dis[start]=0;
    while(!pq.empty())
    {
        TMP = pq.top();
        nowv=TMP.id,nowd=TMP.d;
        dis[nowv]=nowd;
        if(nowv==K) return nowd;
        pq.pop();
        for(int i=0;i<(int)edge[nowv].size();++i)
        {
            nextv = edge[nowv][i].t;
            if(dis[nowv]+edge[nowv][i].w<dis[nextv])
            {
                dis[nextv]=dis[nowv]+edge[nowv][i].w;
                TMP.d=dis[nextv];
                TMP.id=nextv;
                pq.push(TMP);
            }
        }
    }
    return -1;
}



int cmp( const void* p1, const void* p2 ) {
return *(int*)p1 - *(int*)p2;
}
```

Acos(-1.0)=pi

四捨五入 float x; int i; i = x + 0.5

Sort:

```
#Include<cstdlib>
qsprt(起始位置指標,數量,各元素大小,cmp);
int cmp(const void* p1,const void* p2)
[p1<p2(<0) p1==p2(=0) p1>p2(>0)
    return *(int*)p1-*(int*)p2;
#include<algorithm>
```

1.Sort(開始(閉區間),結束(開區間))

```
Bool operator<(const type_name & p)const;
[this<p(true)  this>p(false)]
```

2.sort(start,end,cmp) //遞增

```
bool operator(int p1,int p2)
    return p1>=p2
```

捷徑

x 由小到大排序(若相同看 y),將 x 座標相同的看成一組,若超過 1,則存在捷徑,計算長度比較(兩點相鄰不算)->x,y 互換

bipartitie_matching

```
int bipartite_matching()
{
  Memset(mx,-1,sizeof(mx));
  Memset(my,-1,sizeof(my));
  Int c=ini_matching(); //能連的先連
  For(int x=1;x<nx;++x)
  { if(mx[x]==-1)    //x 為未匹配點
   { memset(vy,false,sizeof(vy));
     If(DNS(x)) c++;
   }
  }
  Return c;
}
Bool DFS(int x)
{
  For(int y=1;y<=ny; ++y)
  { if(adj[x][y] && !vy[y])
   { mx[x]=y;   my[y]=x;
     Return true;
   }
  }
}
```

MaximumFlow

```
int FordFulkerson(int s, int t) {
  int ret = 0;
  while (1) {
    memset(visited, 0, sizeof(visited));
    if (!DFS(s, t))
    break;
    ret += findFlow(s, t);  }
  return ret;  }}
bool DFS(int k, int t) {
  visited[k] = 1;
  if (now == t)
    return 1;
  for (int i = 0; i < n; ++i ) {
    if (visited[i])
      continue;
    if (cap[k][i] - flow[k][i] > 0 || flow[i][k]> 0) {
      path[i] = k;
      if(DFS(i,t))
        return 1;  }  }
  return 0;  }
int findFlow(int s, int t) {
  int f = INF;
  for (int i = t, pre; i != s; i = pre) {
    pre = path[i];
    if (cap[pre][i] - flow[pre][i] > 0)
      f = min(f, cap[pre][i] - flow[pre][i]);
    else
      f = min(f, flow[i][pre]);  }
  for (int i = t, pre; i != s; i = pre) {
    pre = path[i];
    if (cap[pre][i] - flow[pre][i] > 0)
      flow[pre][i] += f;
    else
      flow[i][pre] -= f;  }
return f;  }
```