+ Luồng/Cặp Ghép/Dinitz  
+ Luồng Mincost   
+ Cây It/Bit   
+ Heap   
+ Dequeue Tính Tổng/Min/Max.   
+ Spfa   
+ Euler  
+ Topo Sort   
+ Hình Học Phẳng/Hệ Tọa Độ   
+ Cầu Khớp/Tarjan  
+ Two-Pointer  
+ Xử Lý Bit   
+ Euclide Tìm Ucln/Bcnn  
+ Vét/Duyệt Phân Tập/Giai Thừa/2^N/...  
+ Tất Cả Các Thứ Trong Lmh  
+ Các Thể Loại Sinh Kế Tiếp.   
+ Càng Nhiều Dạng Qhđ Càng Tốt.  
+ Lca (Cả 2 Cách)   
+ Hash

+ Nhân Ma Trận

+ Djistra, Floyd...  
+ Cây Khung (Cả 2 Cách), Tìm Hiểu Thêm Prim Heap.

1. **Dinitz:**

Function Findpath: Boolean;

Var Front,Rear,U,V: Longint;

Tmp: List;

Begin

Fillchar(D,Sizeof(D),0);

D[S]:=1; Front:=1; Rear:=1; Q[1]:=S;

Repeat

U:=Q[Front]; Inc(Front); Tmp:=A[U];

While Tmp<> Nil Do

Begin

V:=Tmp^.V;

If (D[V]=0) And (Fx[U,V]<C[U,V]) Then

Begin

D[V]:=D[U]+1; Inc(Rear); Q[Rear]:=V;

End;

Tmp:=Tmp^.Link; End;

Until Front>Rear;

Exit(D[T]<>0);

End;

Function Incflow(U,Mine: Longint): Longint;

Var Tmp: List; V, Delta: Longint;

Begin

If U = T Then Exit(Mine); Visit[U] := True; Tmp := A[U];

While Tmp <> Nil Do

Begin

V := Tmp^.V;

If Not(Visit[V]) And (Fx[U,V] < C[U,V]) And (D[V] = D[U]+1) Then

Begin Delta := Incflow(V, Min(Mine,C[U,V]-Fx[U,V]));

If Delta <> 0 Then

Begin

Inc(Fx[U,V],Delta); Dec(Fx[V,U],Delta); Exit(Delta); End;

End;

Tmp := Tmp^.Link;

End; Exit(0);

End;

Procedure Try;

Var Tam: Longint;

Begin

Ans:=0;

While Findpath Do

Begin

Fillchar(Visit,Sizeof(Visit),False);

Repeat

Tam:=Incflow(S,Maxlongint); Ans:=Ans+Tam;

Until Tam=0;

End;

Writeln(Ans);

End;

1. **Mincost:**

Function Findpath: Boolean; {Tim Duong Tang Luong Tren Gf Bang Bfs, Tra Ve True Neu Tim Thay}

Var U, V, Front,Rear: Integer;

Queue: Array[1..Max] Of Integer;

Begin

Fillchar(Trace, Sizeof(Trace), 0);

Front := 1; Rear := 1; Queue[1] := S; Trace[S] := N + 1; {Trace[V] = 0 <=> V Chua Tham}

Repeat

U := Queue[Front]; Inc(Front);

For V := 1 To N Do {Xet Cac Dinh V Chua Tham Ke U Tren Gf}

If (Trace[V] = 0) And (C[U, V] > F[U, V]) Then //Xet Cac Cung (U,V) Chua Bao Hoa, Cf=C[U,V]-F[U,V]>0

Begin

Trace[V] := U;

If V = T Then {Den Duoc T Thi Thuat Toan Dung}

Begin Findpath := True; Exit; End;

Inc(Rear); Queue[Rear] := V;

End;

Until Front > Rear;

Findpath := False;

End;

Procedure Incflow; {Tang Luong F Doc Duong Tang Luong P}

Var Delta, U, V: Integer;

Begin

Delta := Maxint;

V := T; {Duyet Duong Tang Luong P Tu Dinh T Den Dinh S}

Repeat

U := Trace[V]; //Xet Cung (U,V) Tren Duong Di P, Voi U La Dinh Tham Truoc V

If C[U, V] - F[U, V] < Delta Then Delta := C[U, V] - F[U, V]; //Tim Gia Tri Cf[U,V] Nho Nhat

V := U;

Until V = S;

V := T; {Duyet Duong Tang Luong P Tu Dinh T Den Dinh S}

Repeat

U := Trace[V]; //Xet Cung (U,V) Tren Duong Di P, Voi U La Dinh Tham Truoc V

F[U, V] := F[U, V] + Delta; F[V, U] := F[V, U] - Delta;

V := U;

Until V = S;

End;

Procedure Printresult;

Var U, V, M: Integer; Fo: Text;

Begin

Assign(Fo, Outputfile); Rewrite(Fo);

M := 0;

For U := 1 To N Do

For V := 1 To N Do

If F[U, V] > 0 Then {Chi Quan Tam Den Nhung Cung Co Luong Duong}

Begin If U = S Then M := M + F[S, V]; End;

Writeln(Fo,M); Close(Fo);

End;

Begin

Enter;

Fillchar(F, Sizeof(F), 0); //Khoi Tao Luong 0

Repeat

If Not Findpath Then Break; //Thoat Neu Khong Tim Thay Duong Tang Luong

Incflow; //Thuc Hien Tang Luong F

Until False;

Printresult; //In Ket Qua

End.

1. **Cây It + Lazy Update: (Cây It Max)**

Procedure Update (X,L,R,U,V : Longint);

Var Mid : Longint;

Begin

Inc(Fx[X],Lazy[X]);

Inc(Lazy[X+X],Lazy[X]); Inc(Lazy[X+X+1],Lazy[X]);

Lazy[X]:=0;

If (U>R) Or (V<L) Then Exit;

If (U<=L) And (R<=V) Then

Begin

Inc(Lazy[X+X],Gt); Inc(Lazy[X+X+1],Gt);

Inc(Fx[X],Gt);

Exit;

End;

Mid:=(L+R) Div 2;

Update(X+X,L,Mid,U,V); Update(X+X+1,Mid+1,R,U,V);

Fx[X]:=Max(Fx[X+X],Fx[X+X+1]);

End;

Procedure Get (X,L,R,U,V : Longint);

Var Mid : Longint;

Begin

Inc(Fx[X],Lazy[X]);

Inc(Lazy[X+X],Lazy[X]); Inc(Lazy[X+X+1],Lazy[X]);

Lazy[X]:=0;

If (U>R) Or (V<L) Then Exit;

If (U<=L) And (R<=V) Then

Begin

Tmp:=Max(Tmp,Fx[X]);

Exit;

End;

Mid:=(L+R) Div 2;

Get(X+X,L,Mid,U,V); Get(X+X+1,Mid+1,R,U,V);

Fx[X]:=Max(Fx[X+X],Fx[X+X+1]);

End;

Begin

Assign(F,Fi); Reset(F); Assign(G,Fo); Rewrite(G);

Readln(F,N,Tong,Test);

For Test:=1 To Test Do

Begin

Readln(F,S,T,Gt);

Tmp:=0;

If S=T Then

Begin Writeln(G,'T'); Continue; End;

Get(1,1,N,S,T-1);

Tmp:=Tong-Tmp;

If Tmp<Gt Then

Begin Writeln(G,'N'); Continue; End;

Writeln(G,'T');

If S=T Then Continue;

Update(1,1,N,S,T-1);

End;

Close(F);

Close(G);

End.

1. **Cây Bit:**

Function Query (X : Longint) : Longint;

Var Sum : Longint;

Begin

Sum:=0;

While X>0 Do

Begin

Inc(Sum,Ft[X]);

X:=X-(X And (-X));

End;

Exit(Sum);

End;

Procedure Update (X,Y : Longint);

Begin

While (X<=Maxi) Do

Begin

Inc(Ft[X],Y);

X:=X+(X And (-X));

End;

End;

**6. Sinh Hoán Vị Kế Tiếp:**

Procedure Xuli;

Var I,J,Vt,Vt1,K:Data;

C:Char;

Begin

Vt:=0;

For I:=Length(S)-1 Downto 1 Do

If Not (S[I]>=S[I+1]) Then Begin Vt:=I; Break; End;

If Vt=0 Then Begin Write('Biggest'); Exit; End;

Vt1:=0;

For I:=Length(S) Downto Vt Do

If S[I]>S[Vt] Then Begin Vt1:=I; Break; End;

If Vt1=0 Then Vt1:=Vt;

C:=S[Vt]; S[Vt]:=S[Vt1]; S[Vt1]:=C;

For I:=1 To Vt Do Write(S[I]);

For I:=Length(S) Downto Vt+1 Do Write(S[I]);

End;

1. **Heap:**

Procedure Swap(Var X,Y:Longint);

Var Temp:Longint;

Begin

Temp:=X; X:=Y; Y:=Temp;

End;

Procedure Upheap(I:Longint);// Neu 1 Nut Lon Hon Nut Cha Cua No Thi Di Chuyen Len Tren

Begin

If (I=1) Or (Heap[I] < Heap[I Div 2]) Then Exit;

Swap(Heap[I],Heap[I Div 2]);

Upheap(I Div 2);

End;

Procedure Downheap(I:Longint);// Neu 1 Nut Nho Hon Nut Con Cua No Thi Di Chuyen Xuong Duoi

Var J:Longint;

Begin

J:=I\*2; If J>Nheap Then Exit;

If (J < Nheap) And (Heap[J] < Heap[J+1]) Then Inc(J);

If Heap[I] < Heap[J] Then

Begin

Swap(Heap[I],Heap[J]);

Downheap(J);

End; End;

Procedure Push(X:Longint); // Dua 1 Phan Tu Moi Vao Heap

Begin

Inc(Nheap);

Heap[Nheap]:=X;

Upheap(Nheap);

End;

Function Pop(V:Longint):Longint;

Begin

Pop:=Heap[V];

Heap[V]:=Heap[Nheap];

Dec(Nheap);

Upheap(V);

Downheap(V);

End; End.

1. **Dequeue Tìm Min/Max Trên Đoạn Tịnh Tiến:**

Procedure Try;

Var I,Lx,Rx: Longint;

Ans: Int64;

Begin

For I:=1 To N Do

Begin L[I]:=I; While (A[L[I]-1]>=A[I]) Do L[I]:=L[L[I]-1]; End;

For I:=N Downto 1 Do

Begin

R[I]:=I; While (A[R[I]+1]>=A[I]) Do R[I]:=R[R[I]+1]; End;

Ans:=0;

For I:=1 To N Do

If Ans<(A[I]\*(R[I]-L[I]+1)) Then

Begin

Ans:=A[I]\*(R[I]-L[I]+1);

Lx:=L[I];

Rx:=R[I];

End;

Writeln(G,Ans,' ',Lx,' ',Rx);

End;

1. **LCA**

***Cách 1: Heavy Light***

Procedure Dfs (X : Longint);

Var Maxc,Y,Vt : Longint;

Begin

Dd[X]:=Dd[Pa[X]]+1;

Child[X]:=1;

Maxc:=0;

For Vt:=1 To Soluong[X] Do

Begin

Y:=Dsk[X,Vt]; If Y=Pa[X] Then Continue;

Pa[Y]:=X; D[Y]:=D[X]+A[X,Y];

Dfs(Y);

Inc(Child[X],Child[Y]);

If Child[Y]>Maxc Then

Begin

Maxc:=Child[Y];

Tree[X]:=Tree[Y];

End;

End;

If Child[X]<>1 Then Exit;

Inc(Ntree);

Tree[X]:=Ntree;

End;

Procedure Init;

Begin

Ntree:=0; Dd[1]:=0;

Pa[1]:=1; Dfs(1);

Dd[0]:=Maxlongint;

For I:=1 To N Do

If Dd[I]<Dd[Root[Tree[I]]] Then Root[Tree[I]]:=I;

End;

Function Lca (X,Y : Longint) : Longint;

Begin

While Tree[X]<>Tree[Y] Do

If Dd[Root[Tree[X]]]>Dd[Root[Tree[Y]]] Then

X:=Pa[Root[Tree[X]]] Else Y:=Pa[Root[Tree[Y]]];

If Dd[X]<Dd[Y] Then Exit(X) Else Exit(Y);

End;

***Cách 2: Spare Table***

Function Log2 (X : Longint) : Longint;

Begin

Exit(Trunc(Ln(X)/Ln(2)));

End;

Procedure Dfs (X : Longint);

Var Chay,Y : Longint;

Begin

For Chay:=1 To Soluong[X] Do

Begin

Y:=Dsk[X,Chay];

If Level[Y]>-1 Then Continue;

Level[Y]:=Level[X]+1;

P[Y]:=X; D[Y]:=D[X]+A[X,Y];

Dfs(Y);

End;

End;

Procedure Init;

Begin

For I:=2 To N Do Level[I]:=-1;

Level[1]:=0; P[1]:=0; D[1]:=0;

Dfs(1);

For I:=1 To N Do Pa[I,0]:=P[I];

For J:=1 To Log2(N) Do

For I:=1 To N Do

If Level[I]-(1 Shl J+1)+1>=0 Then

Pa[I,J]:=Pa[Pa[I,J-1],J-1];

End;

Function Getlca : Longint;

Begin

If Level[U]<Level[V] Then

Begin

I:=U; U:=V; V:=I;

End;

While Level[U]>Level[V] Do

Begin

Tmp:=Log2(Level[U]-Level[V]);

U:=Pa[U,Tmp];

End;

If U=V Then Exit(U);

While Pa[U,0]<>Pa[V,0] Do

Begin

Tmp:=Log2(Level[U]);

For I:=Tmp Downto 0 Do

If Pa[U,I]<>Pa[V,I] Then

Begin

U:=Pa[U,I];

V:=Pa[V,I];

Break;

End;

End;

Exit(Pa[U,0]);

End;

1. **Cầu Khớp:**

Procedure Visit(U:Longint); {Duyet Dfs, Dinh Chieu, Danh So}

Var V:Longint;

Begin

Inc(Count); Number[U] := Count; {Truoc Het Danh So U}

Low[U] := N + 1; {Khoi Gian Low[U] = Oo}

For V := 1 To N Do

If A[U, V] Then {Xet V Ke U}

Begin

A[V, U] := False; {Bo Di Cung (V, U)}

If Parent[V] = 0 Then

{Parent[V] = 0 Tuong Ung V Chua Tham, (U, V) La Cung Thuoc Cay Dfs}

Begin

Parent[V] := U;

Visit(V); {Tham V}

If Low[U] > Low[V] Then Low[U] := Low[V]; {Cuc Tieu Hoa Low[U] Theo Low[V]}

End

Else {V Da Tham, (U, V) La Cung Nguoc}

If Low[U] > Number[V] Then Low[U] := Number[V]; {Cuc Tieu Hoa Low[U] Theo Number[V]}

End;

End;

Procedure Solve;

Var U, V: Longint;

Begin

Count := 0;

{Danh Dau Moi Dinh Deu Chua Tham}

Fillchar(Parent, Sizeof(Parent), 0);

For U := 1 To N Do

If Parent[U] = 0 Then {Gap Mot Dinh Chua Tham}

Begin

Parent[U] := -1; {Cho U La Mot Goc Cay Dfs}

Visit(U); {Xay Dung Cay Dfs Goc U}

End;

End;

Procedure Printresult; //In Ket Qua

Var U, V, Kq1, Kq2: Longint;

Nchildren: Array[1..Max] Of Longint; {Tinh Nchildren[U] = So Nhanh Con Cua Nhanh Dfs Goc U}

Isarticulation: Array[1..Max] Of Boolean; //Xac Dinh Co Phai La Khop Khong

Begin

//Liet Ke Cac Cau

Kq1:=0;Kq2:=0;

For V := 1 To N Do

Begin

U := Parent[V];

If (U <> -1) And (Low[V] >= Number[V]) Then Inc(Kq1);

End;

//Liet Ke Cac Khop

Fillchar(Nchildren, Sizeof(Nchildren), 0);

//Tinh So Nhanh Con Cua Moi Dinh Thuoc Do Thi Sau Khi Dinh Chieu Dfs

For V := 1 To N Do

Begin

U := Parent[V];

If U <> -1 Then Inc(Nchildren[U]);

End;

{Do Thi Vo Huong Ban Dau Co Nhieu Thanh Phan Lien Thong, Tuong Ung Khi Dinh Chieu Dfs

Tao Ra Do Thi Co Huong Voi Nhieu Cay Dfs, Moi Cay Dfs Co Mot Goc}

//U La Khop Neu U La Goc Cay Dfs Va U Co It Nhat 2 Nhanh Con

For U := 1 To N Do

Isarticulation[U]:=(Parent[U] = -1) And (Nchildren[U]>= 2);

//U La Khop Neu U Khong La Goc Cay Dfs Va Low[V]>=Number[U]

For V := 1 To N Do

Begin

U := Parent[V];

If (U <> -1) And (Parent[U] <> -1) Then //Xet Cac Cung (U,V) Voi U Khong Phai La Goc Cay Dfs

If (Low[V] >= Number[U]) Then //Neu Thoa Dieu Kien

Isarticulation[U] := True; //Danh Dau Dinh U La Khop

End;

For U := 1 To N Do //Liet Ke Cac Khop Cua Do Thi

If Isarticulation[U] Then Inc(Kq2);//Writeln(Fo,U);

Writeln(Fo,Kq2,#32,Kq1); End;

**10. Nhân Ma Trận**

Function Nhan(A,B: Matrix;M,N,P:Longint):Matrix;

Var I,J,K: Longint;

C: Matrix;

Begin

Fillchar(C,Sizeof(C),0);

For I:=1 To M Do

For J:=1 To N Do

For K:=1 To P Do

C[I,J]:=(C[I,J]+(A[I,K]\*B[K,J] Mod Base)) Mod Base;

Exit(C);

End;

Function Power(X: Longint):Matrix;

Var C: Matrix;

Begin

C:=B;

If X=1 Then Exit(B);

C:=Nhan(C,C,6,6,6);

If Odd(X) Then C:=Nhan(C,B,6,6,6);

Power(X Div 2);

End;

1. **Kruskal:**

Function Union(I:Longint):Boolean;

Var U,V:Longint;

Begin

U:=List[I].X; V:=List[I].Y;

While U<>T[U] Do U:= T[U];

While V<>T[V] Do V:= T[V];

If(U=V) Then Exit(False);

If(U>V) Then T[U]:=V Else T[V]:=U;

Exit(True);

End;

* For I:=1 To N Do T[I]:=I;

1. **Prim:**

Function Prim:Longint;

Var L,Min: Longint;

Begin

Prim:=0;

For L:=2 To N Do Begin

Min:=Maxlongint;

For I:=2 To N Do

If Q[I] And (D[I]<Min) Then

Begin J:=I; Min:=D[I]; End;

Q[J]:=False;

For I:=2 To N Do

If Q[I] And (D[I]>A[I,J]) Then D[I]:=A[I,J];

Prim:=Prim+D[J];

End;

End;

D[1]:=0;

For N:=2 To N Do

Begin

D[N]:=A[1,N]; Q[N]:=True;

End;

Ans:=Prim;

Assign(F,Fo); Rewrite(F);Writeln(F,Ans); Close(F);

End;

1. **Xử Lí Bit:**

Mọi Số Tự Nhiên Đều Có Thể Biểu Diễn Được Dưới Dạng Nhị Phân. Đối Với Máy Tính, Các Giá Trị Số Được Lưu Bởi Các Biến Dưới Dạng Chuỗi Nhị Phân Có Độ Dài Tương Ứng Với Kiểu Biến Đó.  
Trong Ngôn Ngữ Pascal, Một Số Kiểu Phổ Biến Như:  
 Byte : Độ Dài 8 Bit  
Integer : Độ Dài 2\*8 = 16 Bit  
Longint : Độ Dài 4\*8 = 32 Bit  
Các Bit Của Biến Được Đánh Số Từ Phải Sang Trái Bắt Đầu Từ 1.

**Ưu Điểm Của Xử Lý Bit:**

*1/ Bộ Nhớ:* Xử Lý Bit Có Thể Được Dùng Làm Mảng Đánh Dấu. Thay Vì Sử Dụng 1 Biến Boolean Chỉ Đánh Dấu Được 1 Phần Tử Là True Hay False, Ta Có Thể Xử Lý Bit Để Đánh Dấu 8 Bit Tương Ứng Với 8 Phần Tử.

*2/ Tốc Độ:* Các Phép Xử Lý Bit Có Tốc Độ Nhanh Hơn Các Phép Xử Lý Khác. *Ví Dụ:* Hai Phép (X Div 2) Và (X Shr 1) Là Như Nhau Nhưng Phép (X Shr 1) Có Tốc Độ Nhanh Hơn. Trong Các Bài Toán Đòi Hỏi Việc Thay Đổi Trạng Thái Nhiều Lần Thì Người Ta Vẫn Hay Dùng Xử Lý Bit Để Cải Thiện Tốc Độ Chương Trình.

Sau Đây Là Một Số Phép Xử Lý Bit Cơ Bản:

**1/ Cộng Bit (Or):**  
 Kết Quả Bằng Tổng Giá Trị 2 Bit. Trường Hợp 2 Bit Đều Có Giá Trị Bằng 1 Thì Kết Quả Là 1.  
*Ví Dụ:*  
 X1 = 9 (00001001)  
 X2 = 18 (00010010)  
 X1 Or X2 = 27 (00011011)

**2/ Đảo Bit (Not):**  
 Đảo Bit 0 Thành Bit 1 Và Bit 1 Thành Bit 0.  
*Ví Dụ:*  
 X1= 13 (00001101)  
 Not (X1) = 242 (11110010)

**3/ Nhân Bit (And):**  
 Kết Quả Bằng Tích Giá Trị 2 Bit.  
 *Ví Dụ:*  
 X1= 9 (00001001)  
 X2= 12 (00001100)  
 X1 And X2 = 8 (00001000)

**4/ Loại Trừ Bit (Xor):**  
Kết Quả Là 0 Nếu 2 Bit Có Cùng Giá Trị, Là 1 Nếu 2 Bit Khác Giá Trị.  
*Ví Dụ:*  
X1 = 9  (00001001)  
X2 = 18 (00010010)  
X1 Xor X2 = 27 (00011011)

**5/ Dịch Sang Trái K Bit (Shl):**  
*Ví Dụ:*  
X = 27 (00001101)  
X Shl 2 = 108 (01101100)

**6/ Dịch Sang Phải K Bit (Shr):**  
*Ví Dụ:*X = 27 (00001101)  
X Shr 2 = 6 (00000110)

Từ Các Phép Toán Trên, Ta Có Các Công Thức Mở Rộng Sau:

**7/ Lấy Giá Trị Bit:**  
Cho Biết Bit Thứ K Của Số X Có Giá Trị Bao Nhiêu:

Function Getbit(K , X : Longint) : Longint;  
 Begin  
 Getbit := (X Shr (K-1)) And 1;  
End;

**8/ Gán Giá Trị Bit:**  
Gán Giá Trị C Cho Bit Thứ K Của Số X:

Procedure Setbit(C , K : Longint; Var X : Longint);  
Begin  
 If C = 1 Then X := X Or (1 Shl (K-1) Else X := X And (Not (1 Shl (K-1))); End;

1. **Euler:**

function CanGoBack(x, y: Integer): Boolean;

var Queue: array[1..max] of Integer;

Front, Rear, u, v: Integer;

Free: array[1..max] of Boolean;

begin

Dec(a[x, y]); Dec(a[y, x]);

FillChar(Free, n, True);

Free[y] := False; Front := 1; Rear := 1; Queue[1] := y;

repeat

u := Queue[Front]; Inc(Front);

for v := 1 to n do

if Free[v] and (a[u, v] > 0) then

begin

Inc(Rear); Queue[Rear] := v; Free[v] := False;

if Free[x] then Break;

end;

until Front > Rear;

CanGoBack := not Free[x];

Inc(a[x, y]); Inc(a[y, x]);

end;

procedure FindEulerCircuit;

var Current, Next, v, count: Integer;

f: Text;

begin

Assign(f, OutputFile); Rewrite(f);

Current := 1;

Write(f, 1, ' '); count := 1;

repeat

Next := 0;

for v := 1 to n do

if a[Current, v] > 0 then

begin

Next := v;

if CanGoBack(Current, Next) then Break;

end;

if Next <> 0 then

begin

Dec(a[Current, Next]);

Dec(a[Next, Current]);

Write(f, Next, ' ');

Inc(count); if count mod 16 = 0 then WriteLn;

Current := Next;

end;

until Next = 0;

Close(f);

end;