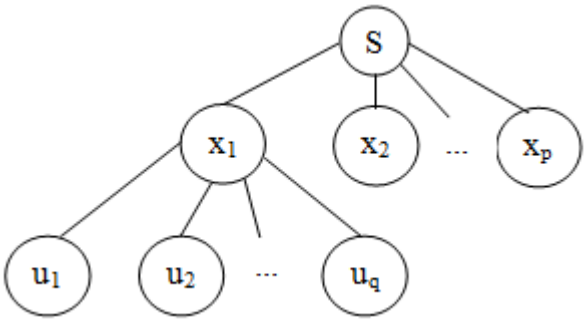
THUẬT TOÁN TÌM KIẾM THEO CHIỀU RỘNG (BFS)

1. Bài toán:

Cho đồ thị G = (V,E) trong đó V là tập các đỉnh và E là tập các cạnh. Tìm đường đi từ s tới t, với s, t $\in$ V.

2. Cài đặt bằng hàng đợi:

Việc thăm một đỉnh sẽ lên lịch duyệt các đỉnh kề nó sao cho thứ tự duyệt là ưu tiên chiều rộng (đỉnh nào gần S hơn sẽ được duyệt trước).



Ví dụ: Bắt đầu thăm đỉnh S. Việc thăm đỉnh S sẽ phát sinh thứ tự duyệt những đỉnh (x_1, x_2, ..., x_p) kề với S. Khi thăm đỉnh x_1 sẽ phát sinh yêu cầu duyệt những đỉnh (u_1,u_2, ..., u_q) kề với x_1. Nhưng các đỉnh u này xa S hơn những đỉnh x nên chúng chỉ được duyệt khi tất cả các đỉnh x đã duyệt xong, nghĩa là thứ tự duyệt đỉnh sau khi đã thăm x_1 sẽ là (x_2, x_3,..., x_p, u_1, u_2,..., u_q)

Giả sử ta có một danh sách chứa những đỉnh đang "chờ" thăm. Tại mỗi bước, ta thăm một đỉnh đầu danh sách và cho những đỉnh chưa "xếp hàng" kề với nó xếp hàng thêm vào cuối danh sách. Chính vì nguyên tắc này nên danh sách chứa những đỉnh đang chờ sẽ được tổ chức dưới dạng hàng đợi.

Giải thuật**:**

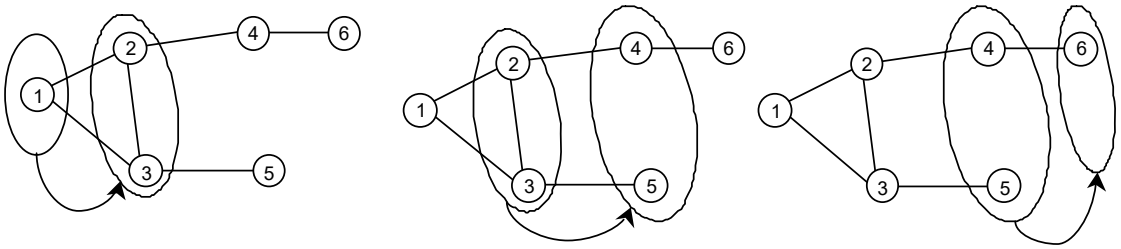
Bước 1: Khởi tạo:
- Các đỉnh đều ở trạng thái chưa đánh dấu, ngoại trừ đỉnh xuất phát S là đã đánh dấu
- Một hàng đợi (Queue), ban đầu chỉ có một phần tử là S. Hàng đợi dùng để chứa các đỉnh sẽ được duyệt theo thứ tự ưu tiên chiều rộng

Bước 2: Lặp các bước sau đến khi hàng đợi rỗng:
- Lấy u khỏi hàng đợi, thông báo thăm u (Bắt đầu việc duyệt đỉnh u)
- Xét tất cả những đỉnh v kề với u mà chưa được đánh dấu, với mỗi đỉnh v đó:
  1. Đánh dấu v
  2. Ghi nhận vết đường đi từ u tới v
  3. Đẩy v vào hàng đợi (v sẽ chờ được duyệt tại những bước sau)

Bước 3: Truy vết đường đi

3. Cài đặt bằng phương pháp loang

Cách cài đặt này sử dụng hai tập hợp, một tập hợp "cũ" chứa những đỉnh "đang xét", một tập hợp "mới" chứa những đỉnh "sẽ xét". Ban đầu, tập hợp cũ chỉ gồm đỉnh xuất phát, tại mỗi bước ta sẽ dùng tập cũ tính tập mới, tập mới sẽ gồm những đỉnh chưa được thăm mà kề với một đỉnh nào đó của tập cũ. Lặp lại công việc trên (sau khi đã gán tập cũ bằng tập mới) cho tới khi tập cũ là rỗng



Giải thuật:

Bước 1: Khởi tạo:
- Các đỉnh khác S đều chưa bị đánh dấu, đỉnh S bị đánh dấu.
- Tập "cũ" Old:={s}

Bước 2: Lặp các bước sau đến khi Old=∅
- Đặt tập "mới" New = ∅.
- Dùng tập "cũ" tính tập "mới" như sau: Xét các đỉnh u ∈Old, với mỗi đỉnh u:
  1. Thông báo thăm u
  2. Xét các đỉnh v kề u mà chưa bị đánh dấu, với mỗi đỉnh v đó:
     - Đánh dấu v
     - Lưu vết đường đi
     - Đưa v vào tập New
  3. Gán tập "cũ" Old := tập "mới" New

Bước 3: Truy tìm đường đi

4. Một số ví dụ minh họa

Bài 1: Bessie rất yêu bãi cỏ của mình và thích thú chạy về chuồng cỏ vào giờ vắt sữa buổi tối. Bessie đã chia đồng cỏ của mình thành một vùng hình chữ nhật gồm các ô vuông nhỏ với R (1<=R<=100) hàng và C (1<=C<=100) cột, đồng thời đánh dấu chỗ nào là cỏ và chỗ nào là đá. Bessie đứng ở vị trí R_b, C_b và muốn ăn cỏ theo cách của mình, từng ô vuông một và trở về chuồng ở ô (1,1); bên cạnh đó đường đi này phải là ngắn nhất.Bessie có thể đi từ 1 ô vuông sang 4 ô vuông khác kề cạnh.

Cho bản đồ với: đá (ký hiệu: '*'), cỏ (ký hiệu: '.'), chuồng bò (ký hiệu: 'B'), Bessie (ký hiệu: 'C'). Hãy tính xem có bao nhiêu ô cỏ mà Bessie sẽ ăn được trên con đường ngắn nhất trở về chuồng (tất nhiên trong chuồng không có cỏ)

Phân tích:

Theo đề bài, Bessie có thể đi từ 1 ô vuông sang 4 ô vuông khác kề cạnh. Đây là mối quan hệ được xây dựng trong bài toán. Khi đó, mỗi ô trong ma trận được xem là 1 đỉnh của đồ thị; nếu từ ô a có thể di chuyển đến ô b thì ta sẽ xây dựng 1 cạnh tương ứng giữa a và b.

Bài toán yêu cầu: Tính số ô cỏ Bessie sẽ ăn trên con đường ngắn nhất từ ô (R_b,C_b) đến ô (1,1).Ta cần xét hết mọi khả năng từ ô (R_b, C_b) có thể đến được ô (1,1), thuật toán dừng khi tới được ô (1,1). Vậy để tìm được kết quả bài toán, ta sẽ áp dụng thuật toán BFS. Với cách tìm ra kết quả như trên, ta có thể cài đặt BFS bằng cả 2 cách: bằng hàng đợi hoặc bằng phương pháp loang.

Mô hình đồ thị của bài toán này như sau:

- Gọi đỉnh của đồ thị tương ứng với mỗi ô (x,y) của ma trận
- Nếu có thể đi từ ô này đến ô kề nó thỏa mãn yêu đề bài (kề cạnh và không phải là ô chưa đá) thì sẽ có một cạnh tương ứng giữa hai đỉnh.
- Bài toán quy về đồ thị như sau: Tìm đường đi ngắn nhất từ đỉnh (R_b,C_b) đến đỉnh (1,1).
- Khi đã giải xong bài toán đồ thị thì đường đi ngắn nhất từ đỉnh (R_b,C_b) đến đỉnh (1,1)là số ô ít nhất Bessie cần đi qua.

Cấu trúc dữ liệu:

- Mảng b[•,•], với b[i,j] cho biết số ô ít nhất đi từ đỉnh (R_b,C_b) đến đỉnh (i,j)
- Mảng a[•,•] – bản đồ đồng cỏ

Mô hình của giải thuật bài toán dùng hàng đợi:

```
Procedure KhoiTao;
Begin
    Fillchar(avail,sizeof(avail),true);
    Fillchar(b,sizeof(b),0);
    Queue:={(R_b,C_b)};
    Avail[R_b,C_b]:=false;
    b[R_b,C_b]:=1;
End;
Function BFS:word;
Begin
   Repeat
     U:=pop;
     <Xét 4 đỉnh v kề u chưa được đưa vào hàng đợi, với mỗi đỉnh v:>
     Begin
         avail[v.x,v.y]:=false;
         If a[v.x,v.y]='.' then
         Begin
             b[v.x,v.y]:=b[u.x,u.y]+1;
             Push(v);
         End;
         If a[v.x,v.y]='B' then
         Begin
             b[v.x,v.y]:=b[u.x,u.y];
             Exit(b[v.x,v.y]);
         End;
     End;
    Until Queue=∅;
End;
Procedure Process;
Begin
    KhoiTao;
    Write(BFS);
End;
```

Bài 2:Vào mùa hè, trời nóng nên chú bò Bessie rất lười. Bessie muốn tìm vị trí đứng trong nông trại của mình sao cho tại vị trí ấy Bessi có thể ăn được nhiều cỏ nhất có thể. Nông trại của Bessie được mô tả bằng một ma trận N x N (1 <= N <= 400). Ô ỏ dòng r và cột c (1 <= r, c <= N) chứa

G(r,c) đơn vị cỏ (0 <= G(r,c) <= 1000). Tại vị trí ban đầu của mình, Bessie chỉ muốn di chuyển K bước (0 <= K <= 2*N). Mỗi bước, Bessie di chuyển đến ô nằm ở hướng Đông, Tây, Nam và Bắc.Ví dụ, giả sử với ma trận sau, vị trí xuất phát của Bessie nằm ở dòng 3, cột 3.

50	5	**25**	6	17
14	**3**	**2**	**7**	21
99	**10**	**1**	**2**	**80**
8	**7**	**5**	**23**	11
10	0	**78**	1	9

Nếu K = 2 thì Bessie có thể đi vào các ô được tô.

Hãy giúp Bessie xác định tổng lượng cỏ lớn nhất mà Bessie có thể ăn, nếu đặt trường hợp cô ấy chọn vị trí xuất phát là tốt nhất.

Phân tích

Theo đề bài: Có N^2 cách chọn vị trí xuất phát. Với mỗi cách chọn, ta cần tìm tổng lượng cỏ Bessi có thể ăn trong phạm vi K bước. Vị trí xuất phát thuận lợi nhất là vị trí có tổng lượng cỏ lớn nhất. Với mỗi cách chọn, ta cần tính tổng lượng cỏ Bessie có thể ăn trong phạm vi K bước. Nhìn vào ví dụ: Với K=2, ta chọn ô xuất phát là ô (3,3) thì: Bước 0: Xét ô (3,3); Bước 1: Ta sẽ xét các ô kề với ô ở bước 0, đó là các ô: (2,3); (3,2); (4,3); (3,4); Bước 2: Ta sẽ xét các ô kề với các ô ở bước 1, đó là các ô: (1,3); (2,2); (2,4); (3,1); (4,2); (5,3); (4,4); (3,5).Các ô được xét ở bước thứ i chính là các ô được phát sinh từ các ô ở bước thứ i-1. Do đó, ta áp dụng phương pháp loang để tính tổng lượng cỏ Bessie có thể ăn trong phạm vi K bước. Để áp dụng phương pháp loang, ta cần đưa về bài toán đồ thị.

Theo đề bài, mỗi bước, Bessie di chuyển đến ô nằm ở hướng Đông, Tây, Nam và Bắc. Đây là mối quan hệ được xây dựng trong bài toán. Khi đó, mỗi ô trong ma trận được xem là 1 đỉnh của đồ thị; nếu từ ô a có thể di chuyển đến ô b thì ta sẽ xây dựng 1 cạnh tương ứng giữa a và b.

Mô hình đồ thị của bài toán như sau:

- Gọi đỉnh của đồ thị tương ứng với mỗi ô của ma trận
- Nếu có thể đi từ ô này đến ô kề nó thỏa mãn yêu đề bài (kề cạnh) thì sẽ có một cạnh tương ứng giữa hai đỉnh.
- Bài toán quy về dạng đồ thị như sau: Với mỗi cách chọn đỉnh xuất phát (x,y), hãy tính tổng giá trị các đỉnh đi qua bằng phương pháp loang.
  Gọi sum_i: là tổng giá trị các đỉnh đi qua ứng với cách chọn đỉnh xuất phát
- Khi đã giải xong bài toán đồ thị thì kết quả cần tìm chính là: $Max\{sum_i, \forall i = 1,2,\ldots,N^2\}$.

Cấu trúc dữ liệu

- Mảng a[•,•] – nông trại của Bessie
- sum:tổng lượng cỏ Bessie có thể ăn nếu vị trí xuất phát là ô (i,j)
- kq: tổng lượng cỏ lớn nhất Bessie có thể ăn.

Mô hình giải thuật của bài toán

```
Procedure KhoiTao(i,j);
Begin
    Fillchar(avail,sizeof(avail),true);
```

```
    Old:={(i,j)};
    Avail[i,j]:=false;
    Sum:=a[i,j];
End;
Procedure BFS;
Begin
   Dem:=0;
   Repeat
     Inc(dem);
     New:=∅;
     For ∀(u,v)∈ Old do
     Begin
         <Xét 4 đỉnh (x,y) kề đỉnh (u,v) chưa được đưa vào hàng đợi, với mỗi
         đỉnh (x,y):>
         Begin
             avail[x,y]:=false;
             inc(sum,a[x,y]);
             New:=New+{(x,y)};
         End;
     End;
     Old:=New;
   Until (dem=k) or (Old=∅);
End;
Procedure Process;
Begin
    Kq:=0;
    For x ∈ [1,n] do
        For y ∈ [1,n] do
        Begin
             KhoiTao(x,y);
             BFS;
             Kq:=max(kq,sum);
        End;
    Write(kq);
End;
```

Bài 3:Mỗi một số nguyên dương đều có thể biểu diễn dưới dạng tích của 2 số nguyên dương X, Y sao cho X ≤ Y. Nếu như trong phân tích này ta thay X bởi X-1 còn Y bởi Y+1 thì sau khi tínhtích của chúng ta thu được hoặc là một số nguyên dương mới hoặc là số 0.Ví dụ: số 12 có 3 cách phân tích 1*12, 3*4, 2*6. Cách phân tích thứ nhất cho ta tích mới là 0:(1-1)*(12+1) = 0, cách phân tích thứ hai cho ta tích mới là 10: (3-1)*(4+1) =10, còn cáchphân tích thứ ba cho ta 7: (2-1)*(6+1) = 7. Nếu như kết quả là khác không ta lại lặp lại thủtục này đối với số thu được. Rõ ràng áp dụng liên tiếp thủ tục trên, cuối cùng ta sẽ đến đượcsố 0, không phụ thuộc vào việc ta chọn cách phân tích nào để tiếp tục.Cho trước số nguyên dương N (1 ≤ N ≤ 10000), hãy đưa ra tất cả các số nguyên dương khác nhau có thể gặp trong việc áp dụng thủ tục đã mô tả đối với N.

Phân tích:

Nhận xét việc phân tích số 12, ta sẽ tìm được thêm 2 số là số 7 và số 10. Lấy số 7 ra xét, ta không tìm được thêm số nào mới nữa. Lấy số 10 ra xét, ta tìm được thêm số 6. Lấy số 6 ra xét ta

tìm được thêm số 4. Lấy số 4 ra xét, ta tìm được thêm số 3. Lấy số 3 ra xét, ta không tìm được thêm số nào nữa.

Xét số N, với cách phân tích số N theo đề bài, ta có thể tìm thêm được các số mới. Ta cũng lần lượt xét các số này để tìm xem thử còn số mới nào nữa không? Việc này lặp đi lặp lại cho tới khi không còn số mới nào nữa. Với mối quan hệ này, mỗi con số được sinh ra tương ứng với một đỉnh trong đồ thị, nếu việc phân tích số a sinh ra được số b thì sẽ có một cạnh giữa 2 đỉnh a và b. Với cách phân tích này, ta có thể cài đặt BFS bằng cả 2 cách. Thuật toán dừng khi không còn phát sinh được thêm số nào nữa.

Mô hình đồ thị của bài toán như sau:

- Gọi đỉnh của đồ thị ứng với một số nguyên dương mới được tìm được. Khi đó, đỉnh xuất phát S tương ứng với số nguyên dương N.
- Xây dựng cạnh tương ứng giữa đỉnh a và đỉnh b nếu khi phân tích số a theo cách của bài toán ta tìm được số nguyên dương b mới.
- Bài toán quy về dạng đồ thị như sau: Đồ thị G ban đầu chỉ có 1 đỉnh S. Từ đỉnh S, xây dựng thêm các đỉnh của đồ thị.
- Khi đã giải xong bài toán đồ thị thì kết quả cần tìm chính là các đỉnh trong đồ thị.

Mô hình giải thuật của bài toán

```
Procedure KhoiTao;
Begin
    Fillchar(avail,sizeof(avail),true);
    Queue:={N};
End;
Procedure BFS;
Begin
   Repeat
     U := Pop;
     For ∀x∈[2,√u] do
         If (u mod x=0) and avail[(x-1)*((u div x)+1)] then
             Push((x-1)*((u div x)+1));
   Until Queue=∅;
End;
Procedure Process;
Begin
    KhoiTao;
    BFS;
    For ∀x∈Queue do write(x,' ');
End;
```

HẾT