HUST

ĐẠI HỌC BÁCH KHOA HÀ NỘI

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

ONE LOVE. ONE FUTURE.

APPLIED ALGORITHMS

ĐẠI HỌC
BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY
OF SCIENCE AND TECHNOLOGY

APPLIED ALGORITHMS

DEPTH FIRST SEARCH (DFS) AND APPLICATIONS

ONE LOVE. ONE FUTURE.

CONTENTS

- The longest path on a tree (Đường đi dài nhất trên cây)
- Total path length on the tree (Tổng đường đi trên cây)

THE LONGEST PATH ON THE TREE

- Given a tree $T = (V, E)$, each edge (u,v) has weight $w(u,v)$. Find the path with the longest length on T (the length of the path is the sum of weight on all edges of the path).
- Denote $A[v]$ is the set of vertices adjacent to vertex v on T
- The algorithm is based on depth-first-search (DFS)
  - Choose an arbitrary vertex s on T
  - Perform DFS(s) to find vertex x farthest from s
  - Perform DFS(x) to find the vertex y that is farthest from x
  - The path from x to y found will be the longest path on T

THE LONGEST PATH ON THE TREE

```
Init(V, A) {
   for v in V do d[v] = -1;
}
DFS(u) {
  for x in A[u] do {
    if d[x] < 0 then {
       d[x] = d[u] + w(u,x);
       DFS(x);
    }
  }
}
```

```
LongestPathOnTree(V, A){
  Init(V, A);
  s = select a node in V;
  DFS(s);
  x = select u in V such that d[u] is maximal;
  Init(V, A);
  DFS(x);
  y = select u in V such that d[u] is maximal;
  P = unique path between x and y in T;
  return P;
}
```

THE LONGEST PATH ON THE TREE

- The complexity: $O(|V| + |E|)$

TOTAL PATH LENGTH ON THE TREE

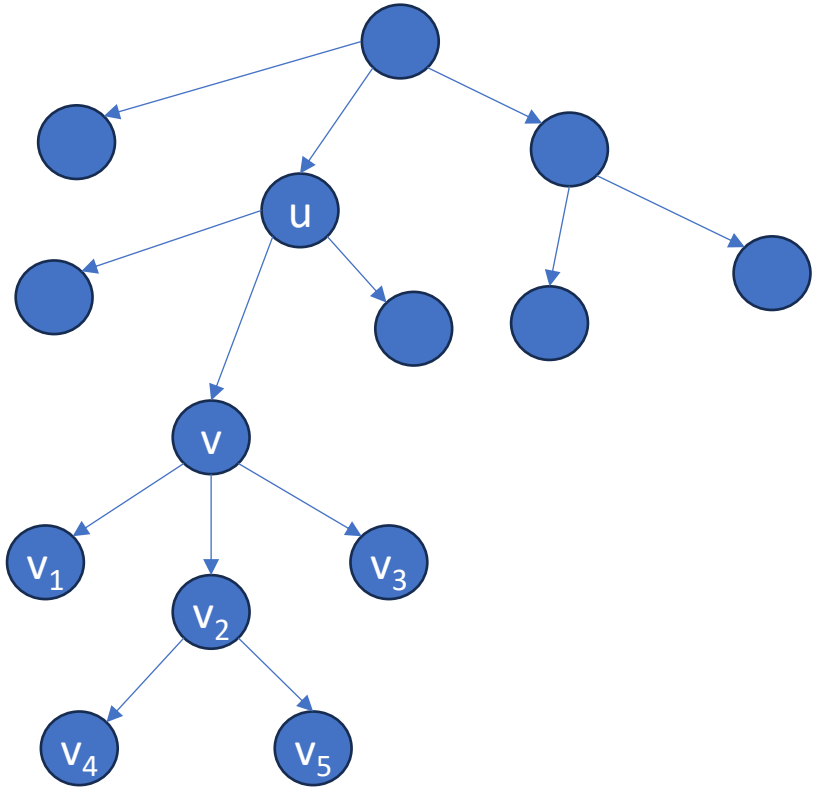
- Given a tree $T = (V, E)$, each edge (u,v) has weight $w(u,v)$. Vertex set V includes n vertices.
- Denote:
  - $A[v]$: is the set of vertices adjacent to vertex v on T
  - $c(u,v)$ is the length of the unique path between two vertices u and v on T
  - $f(u)$: total path length from other vertices to u on T: $f(u) = \sum_{v \in V} c(v,u)$
- Find $f(u)$ for every $u \in V$

TOTAL PATH LENGTH ON THE TREE

- Choose an arbitrary vertex s on T as the root, perform DFS on T starting from s:
  - $p(u)$: parent vertex of u (the vertex from which the algorithm visits u)
  - $d(u)$: total path length from descendant vertices of u to u
  - $N(u)$: number of descendant vertices of u (including vertex u)

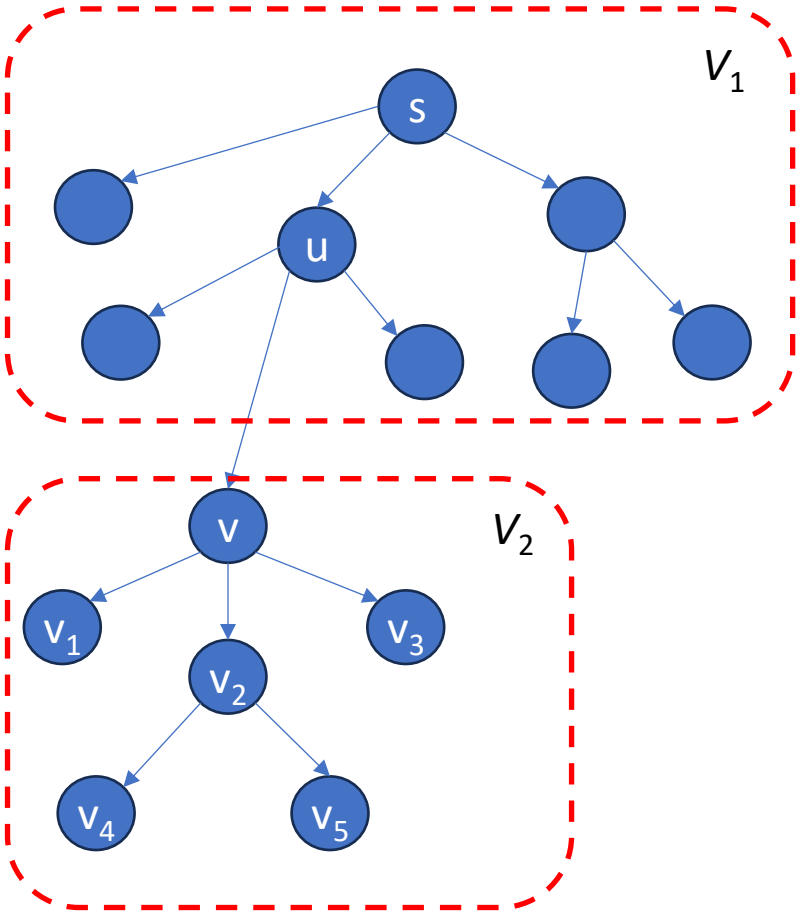
TOTAL PATH LENGTH ON THE TREE

- DFS1(u): depth-first search in the first phase
  - Purpose: calculate $d(x)$ and $N(x)$ for all vertices x that are descendants of u
  - When DFS1(u) is completed, $d(u)$ is calculated and it will be used to calculate $d(p(u))$
  - Do: for each vertex $v \in A[u]$:
    - Call DFS1(v)
    - Update: $d(u) = d(u) + d(v) + N(v)^*w(u,v)$
    - $N(u) = N(u) + N(v)$



TOTAL PATH LENGTH ON THE TREE

- DFS1(u): depth-first search in the first phase
  - Purpose: calculate $d(x)$ and $N(x)$ for all vertices x that are descendants of u
  - When DFS1(u) is completed, $d(u)$ is calculated and it will be used to calculate $d(p(u))$
  - Do: for each vertex $v \in A[u]$:
    - Call DFS1(v)
    - Update: $d(u) = d(u) + d(v) + N(v)*w(u,v)$
    - $N(u) = N(u) + N(v)$
- DFS2(u): depth-first search in the second phase
  - Purpose: When DFS2(u) is called, $f(u)$ has been already calculated and we will calculate $f(v)$ for each vertex v being a child of u
  - Do: for each vertex $v \in A[u]$ not has been visited
    - $F = f(u) - (d(v) + w(u,v)*N(v))$
    - $f(v) = F + d(v) + w(u,v)*(n - N(v))$
    - Call DFS2(v)



TOTAL PATH LENGTH ON THE TREE

```
DFS1(u){
  for v in A[u] do {
     if p(v) = 0 then {
        p(v) = u;
        DFS1(v);
        d(u) = d(u) + d(v) + N(v)*w(u,v);
        N(u) = N(u) + N(v);
     }
  }
}
Phase1(){
  for v in V do {
     p(v) = 0; d(v) = 0; N(v) = 1; f(v) = 0;
  }
  p(1) = 1; DFS1(1);
}
```

```
DFS2(u){
  for v in A[u] do {
     if p(v) = 0 then {
        F = f(u) – (d(v) + N(v)*w(u,v));
        f(v) = F + d(v) + w(u,v)*(n – N(v));
        p(v) = u; DFS2(v);
     }
  }
}
Phase2(){
  for v in V do { p(v) = 0; }
  f(1) = d(1); p(1) = 1; DFS2(1);
}
Main(){
   Phase1(); Phase2();
}
```

TOTAL PATH LENGTH ON THE TREE

- The complexity: $O(|V| + |E|)$

HUST

hust.edu.vn fb.com/dhbkhn

THANK YOU !