CS F364: Design & Analysis of Algorithms

Lecture-kt17: Fibonacci Heap (contd..) + Graph Algorithms

Dr. Kamlesh Tiwari,
Assistant Professor,
Department of Computer Science and Information Systems,
BITS Pilani, Rajasthan-333031 INDIA

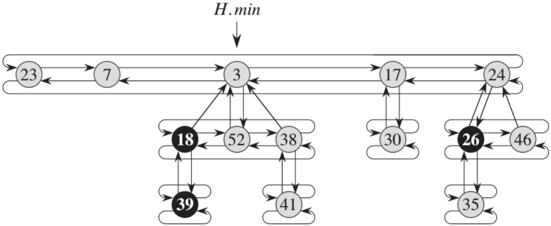
Mar 04, 2017 (Campus @ BITS-Pilani Jan-May 2017)

Fibonacci Heaps (contd..)

- A **Potential function** for Fibonacci Heap is defined as $\Phi(H) = t(H) + 2m(H)$ where $t(H)$ is trees in root-list and $m(H)$ is number of black node

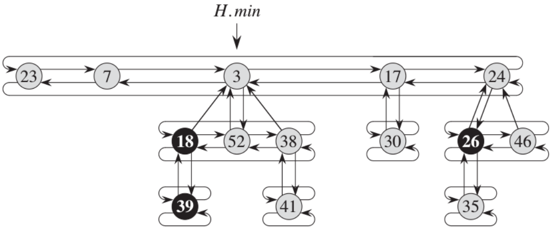
Fibonacci Heaps (contd..)

- A **Potential function** for Fibonacci Heap is defined as $\Phi(H) = t(H) + 2m(H)$ where $t(H)$ is trees in root-list and $m(H)$ is number of black node
- For following FH



Fibonacci Heaps (contd..)

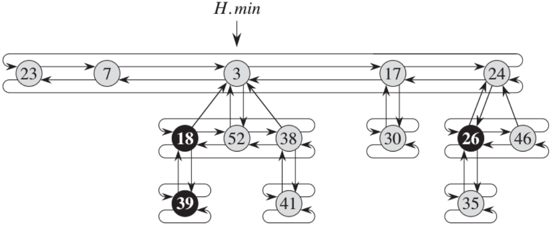
- A **Potential function** for Fibonacci Heap is defined as $\Phi(H) = t(H) + 2m(H)$ where $t(H)$ is trees in root-list and $m(H)$ is number of black node
- For following FH



- $\Phi(H) = 5 + 2 \times 3 = 11$

Fibonacci Heaps (contd..)

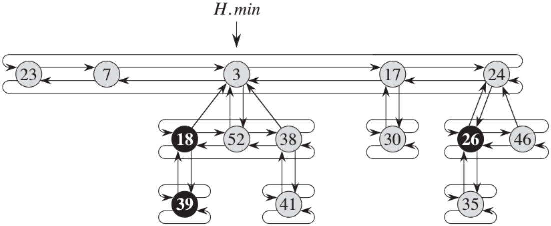
- A **Potential function** for Fibonacci Heap is defined as $\Phi(H) = t(H) + 2m(H)$ where $t(H)$ is trees in root-list and $m(H)$ is number of black node
- For following FH



- $\Phi(H) = 5 + 2 \times 3 = 11$
- Unit of potential can pay for a constant amount of work (sufficiently large to cover the cost of any constant-time pieces of work)

Fibonacci Heaps (contd..)

- A **Potential function** for Fibonacci Heap is defined as $\Phi(H) = t(H) + 2m(H)$ where $t(H)$ is trees in root-list and $m(H)$ is number of black node
- For following FH



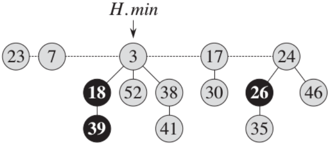
- $\Phi(H) = 5 + 2 \times 3 = 11$
- Unit of potential can pay for a constant amount of work (sufficiently large to cover the cost of any constant-time pieces of work)
- Let $D(n)$ be the the maximum degree of any node in an n-node Fibonacci heap

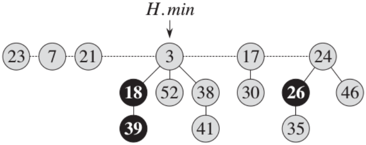
Fibonacci Heaps (contd..)

- Creating a new Fibonacci heap: $\Phi(H) = 0$

Fibonacci Heaps (contd..)

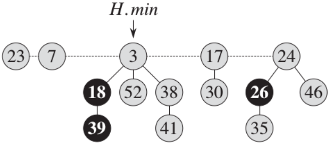
- Creating a new Fibonacci heap: $\Phi(H) = 0$
- Inserting a node: $\Phi(H) == \Phi(H) + 1$

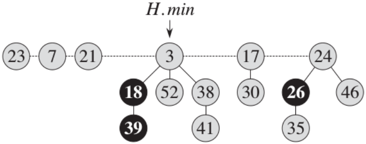




Fibonacci Heaps (contd..)

- Creating a new Fibonacci heap: $\Phi(H) = 0$
- Inserting a node: $\Phi(H) == \Phi(H) + 1$





- Uniting two Fibonacci heaps: $\Phi(H) == \Phi(H_1) + \Phi(H_2)$

Extracting the minimum node

```
CONSOLIDATE(H)
let A[0 . . D(H.n)] be a new array
for i = 0 to D(H.n)
    A[i] = NIL
for each node w in the root list of H
    x = w
    d = x.degree
    while A[d] ≠ NIL
        y = A[d]          // another node with the same degree as x
        if x.key > y.key
            exchange x with y
        FIB-HEAP-LINK(H, y, x)
        A[d] = NIL
        d = d + 1
    A[d] = x
H.min = NIL
for i = 0 to D(H.n)
    if A[i] ≠ NIL
        if H.min == NIL
            create a root list for H containing just A[i]
            H.min = A[i]
        else insert A[i] into H's root list
            if A[i].key < H.min.key
                H.min = A[i]
```

```
FIB-HEAP-LINK(H, y, x)
remove y from the root list of H
make y a child of x, incrementing x.degree
y.mark = FALSE
```

```
FIB-HEAP-EXTRACT-MIN(H)
z = H.min
if z ≠ NIL
    for each child x of z
        add x to the root list of H
        x.p = NIL
    remove z from the root list of H
    if z == z.right
        H.min = NIL
    else H.min = z.right
        CONSOLIDATE(H)
    H.n = H.n − 1
return z
```

Extracting the minimum node

```
CONSOLIDATE(H)
let A[0..D(H.n)] be a new array
for i = 0 to D(H.n)
    A[i] = NIL
for each node w in the root list of H
    x = w
    d = x.degree
    while A[d] ≠ NIL
        y = A[d]          // another node with the same degree as x
        if x.key > y.key
            exchange x with y
        FIB-HEAP-LINK(H, y, x)
        A[d] = NIL
        d = d + 1
    A[d] = x
H.min = NIL
for i = 0 to D(H.n)
    if A[i] ≠ NIL
        if H.min == NIL
            create a root list for H containing just A[i]
            H.min = A[i]
        else insert A[i] into H's root list
            if A[i].key < H.min.key
                H.min = A[i]
```

```
FIB-HEAP-LINK(H, y, x)
remove y from the root list of H
make y a child of x, incrementing x.degree
y.mark = FALSE
```

```
FIB-HEAP-EXTRACT-MIN(H)
z = H.min
if z ≠ NIL
    for each child x of z
        add x to the root list of H
        x.p = NIL
    remove z from the root list of H
    if z == z.right
        H.min = NIL
    else H.min = z.right
        CONSOLIDATE(H)
    H.n = H.n − 1
return z
```

- Potential before extraction is $t(H) + 2m(H)$

Extracting the minimum node

```
CONSOLIDATE(H)
let A[0..D(H.n)] be a new array
for i = 0 to D(H.n)
    A[i] = NIL
for each node w in the root list of H
    x = w
    d = x.degree
    while A[d] ≠ NIL
        y = A[d]        // another node with the same degree as x
        if x.key > y.key
            exchange x with y
        FIB-HEAP-LINK(H, y, x)
        A[d] = NIL
        d = d + 1
    A[d] = x
H.min = NIL
for i = 0 to D(H.n)
    if A[i] ≠ NIL
        if H.min == NIL
            create a root list for H containing just A[i]
            H.min = A[i]
        else insert A[i] into H's root list
            if A[i].key < H.min.key
                H.min = A[i]
```

```
FIB-HEAP-LINK(H, y, x)
remove y from the root list of H
make y a child of x, incrementing x.degree
y.mark = FALSE
```

```
FIB-HEAP-EXTRACT-MIN(H)
z = H.min
if z ≠ NIL
    for each child x of z
        add x to the root list of H
        x.p = NIL
    remove z from the root list of H
    if z == z.right
        H.min = NIL
    else H.min = z.right
        CONSOLIDATE(H)
    H.n = H.n − 1
return z
```

- Potential before extraction is $t(H) + 2m(H)$
- After it is at most $(D(n) + 1) + 2m(H)$

Extracting the minimum node

CONSOLIDATE(H)

```
let A[0 . . D(H.n)] be a new array
for i = 0 to D(H.n)
    A[i] = NIL
for each node w in the root list of H
    x = w
    d = x.degree
    while A[d] ≠ NIL
        y = A[d]          // another node with the same degree as x
        if x.key > y.key
            exchange x with y
        FIB-HEAP-LINK(H, y, x)
        A[d] = NIL
        d = d + 1
    A[d] = x
H.min = NIL
for i = 0 to D(H.n)
    if A[i] ≠ NIL
        if H.min == NIL
            create a root list for H containing just A[i]
            H.min = A[i]
        else insert A[i] into H's root list
            if A[i].key < H.min.key
                H.min = A[i]
```

FIB-HEAP-LINK(H, y, x)

```
remove y from the root list of H
make y a child of x, incrementing x.degree
y.mark = FALSE
```

FIB-HEAP-EXTRACT-MIN(H)

```
z = H.min
if z ≠ NIL
    for each child x of z
        add x to the root list of H
        x.p = NIL
    remove z from the root list of H
    if z == z.right
        H.min = NIL
    else H.min = z.right
        CONSOLIDATE(H)
    H.n = H.n − 1
return z
```

- Potential before extraction is $t(H) + 2m(H)$
- After it is at most $(D(n) + 1) + 2m(H)$
- So $O(D(n))$

Bounding Maximum Degree

Lemma-01

Let $y_1, y_2, ..., y_{x.degree}$ be the children of x in order they are linked then $y_i.degree \geq i - 2$ for $i > 1$

Bounding Maximum Degree

Lemma-01

Let $y_1, y_2, ..., y_{x.degree}$ be the children of x in order they are linked then $y_i.degree \geq i - 2$ for $i > 1$

Proof:

- y_i was linked when $y_1, y_2, ..., y_{i-1}$ were already there so, $x.degree \geq i - 1$

Bounding Maximum Degree

Lemma-01

Let $y_1, y_2, ..., y_{x.degree}$ be the children of x in order they are linked then $y_i.degree \geq i - 2$ for $i > 1$

Proof:

- y_i was linked when $y_1, y_2, ..., y_{i-1}$ were already there so, $x.degree \geq i - 1$
- But, y_i and x can only be linked if their degree be same

Bounding Maximum Degree

Lemma-01

Let $y_1, y_2, ..., y_{x.degree}$ be the children of x in order they are linked then $y_i.degree \geq i - 2$ for $i > 1$

Proof:

- y_i was linked when $y_1, y_2, ..., y_{i-1}$ were already there so, $x.degree \geq i - 1$
- But, y_i and x can only be linked if their degree be same
- Therefore, at the time of linkage $y_i.degree \geq i - 1$

Bounding Maximum Degree

Lemma-01

Let $y_1, y_2, ..., y_{x.degree}$ be the children of x in order they are linked then $y_i.degree \geq i - 2$ for $i > 1$

Proof:

- y_i was linked when $y_1, y_2, ..., y_{i-1}$ were already there so, $x.degree \geq i - 1$
- But, y_i and x can only be linked if their degree be same
- Therefore, at the time of linkage $y_i.degree \geq i - 1$
- Since then it may have lost one child so

Bounding Maximum Degree

Lemma-01

Let $y_1, y_2, ..., y_{x.degree}$ be the children of x in order they are linked then $y_i.degree \geq i - 2$ for $i > 1$

Proof:

- y_i was linked when $y_1, y_2, ..., y_{i-1}$ were already there so, $x.degree \geq i - 1$
- But, y_i and x can only be linked if their degree be same
- Therefore, at the time of linkage $y_i.degree \geq i - 1$
- Since then it may have lost one child so
- $y_i.degree \geq i - 2$

Bounding Maximum Degree

Fibonacci Sequence: $F_0 = 0$, $F_1 = 1$, $F_n = F_{n-1} + F_{n-2}$

Bounding Maximum Degree

Fibonacci Sequence: $F_0 = 0$, $F_1 = 1$, $F_n = F_{n-1} + F_{n-2}$

Lemma-02

For $k \geq 0$,

$$F_{k+2} = 1 + \sum_{i=0}^{k} F_i$$

Bounding Maximum Degree

Fibonacci Sequence: $F_0 = 0$, $F_1 = 1$, $F_n = F_{n-1} + F_{n-2}$

Lemma-02

For $k \geq 0$,

$$F_{k+2} = 1 + \sum_{i=0}^{k} F_i$$

Proof: Use induction (Base case, for $k = 0$, is trivial)

Bounding Maximum Degree

Fibonacci Sequence: $F_0 = 0$, $F_1 = 1$, $F_n = F_{n-1} + F_{n-2}$

Lemma-02

For $k \geq 0$,

$$F_{k+2} = 1 + \sum_{i=0}^{k} F_i$$

Proof: Use induction (Base case, for $k = 0$, is trivial)

$$F_{k+2}$$

Bounding Maximum Degree

Fibonacci Sequence: $F_0 = 0$, $F_1 = 1$, $F_n = F_{n-1} + F_{n-2}$

Lemma-02

For $k \geq 0$,

$$F_{k+2} = 1 + \sum_{i=0}^{k} F_i$$

Proof: Use induction (Base case, for $k = 0$, is trivial)

$$\begin{aligned} F_{k+2} & \\ &= F_k + F_{k+1} \end{aligned}$$

Bounding Maximum Degree

Fibonacci Sequence: $F_0 = 0$, $F_1 = 1$, $F_n = F_{n-1} + F_{n-2}$

Lemma-02

For $k \geq 0$,

$$F_{k+2} = 1 + \sum_{i=0}^{k} F_i$$

Proof: Use induction (Base case, for $k = 0$, is trivial)

$$\begin{aligned} & F_{k+2} \\ = \quad & F_k + F_{k+1} \\ = \quad & F_k + \left(1 + \sum_{i=0}^{k-1} F_i\right) \end{aligned}$$

Bounding Maximum Degree

Fibonacci Sequence: $F_0 = 0$, $F_1 = 1$, $F_n = F_{n-1} + F_{n-2}$

Lemma-02

For $k \geq 0$,

$$F_{k+2} = 1 + \sum_{i=0}^{k} F_i$$

Proof: Use induction (Base case, for $k = 0$, is trivial)

$$\begin{aligned} &F_{k+2} \\ &= F_k + F_{k+1} \\ &= F_k + \left(1 + \sum_{i=0}^{k-1} F_i\right) \\ &= 1 + \sum_{i=0}^{k} F_i \end{aligned}$$

Bounding Maximum Degree

Golden ration: $\Phi = (1 + \sqrt(5))/2$ is root of the equation $x^2 = x + 1$

Lemma-03

Bounding Maximum Degree

Golden ration: $\Phi = (1 + \sqrt(5))/2$ is root of the equation $x^2 = x + 1$

Lemma-03

$$F_{k+2} \geq \Phi^k$$

Bounding Maximum Degree

Golden ration: $\Phi = (1 + \sqrt{(5)})/2$ is root of the equation $x^2 = x + 1$

Lemma-03

$$F_{k+2} \geq \Phi^k$$

Proof: Use induction (Base case, for $k = 0$, is trivial)

Bounding Maximum Degree

Golden ration: $\Phi = (1 + \sqrt(5))/2$ is root of the equation $x^2 = x + 1$

Lemma-03

$$F_{k+2} \geq \Phi^k$$

Proof: Use induction (Base case, for $k = 0$, is trivial)

$$F_{k+2}$$

Bounding Maximum Degree

Golden ration: $\Phi = (1 + \sqrt{(5)})/2$ is root of the equation $x^2 = x + 1$

Lemma-03

$$F_{k+2} \geq \Phi^k$$

Proof: Use induction (Base case, for $k = 0$, is trivial)

$$\begin{aligned} & F_{k+2} \\ = \quad & F_{k+1} + F_k \end{aligned}$$

Bounding Maximum Degree

Golden ration: $\Phi = (1 + \sqrt(5))/2$ is root of the equation $x^2 = x + 1$

Lemma-03

$$F_{k+2} \geq \Phi^k$$

Proof: Use induction (Base case, for $k = 0$, is trivial)

$$\begin{aligned} & F_{k+2} \\ = \quad & F_{k+1} + F_k \\ \geq \quad & \Phi^{k-1} + \Phi^{k-2} \end{aligned}$$

Bounding Maximum Degree

Golden ration: $\Phi = (1 + \sqrt(5))/2$ is root of the equation $x^2 = x + 1$

Lemma-03

$$F_{k+2} \geq \Phi^k$$

Proof: Use induction (Base case, for $k = 0$, is trivial)

$$\begin{aligned}
& F_{k+2} \\
& \quad = F_{k+1} + F_k \\
& \quad \geq \Phi^{k-1} + \Phi^{k-2} \\
& \quad = \Phi^{k-2}(\Phi + 1) = \Phi^{k-2}(\Phi^2)
\end{aligned}$$

Bounding Maximum Degree

Golden ration: $\Phi = (1 + \sqrt{(5)})/2$ is root of the equation $x^2 = x + 1$

Lemma-03

$$F_{k+2} \geq \Phi^k$$

Proof: Use induction (Base case, for $k = 0$, is trivial)

$$\begin{aligned}
& F_{k+2} \\
& \quad = \quad F_{k+1} + F_k \\
& \quad \geq \quad \Phi^{k-1} + \Phi^{k-2} \\
& \quad = \quad \Phi^{k-2}(\Phi + 1) = \Phi^{k-2}(\Phi^2) \\
& \quad = \quad \Phi^k
\end{aligned}$$

Bounding Maximum Degree

Lemma-04

$size(x) \geq F_{k+2} \geq \Phi^k$ where $k = x.dergee$ for an node x in FH

Bounding Maximum Degree

Lemma-04

$size(x) \geq F_{k+2} \geq \Phi^k$ where $k = x.dergee$ for an node x in FH

Proof: Let s_k be minimum size of a node having degree k. (Bases cases are $s_0 = 1$, $s_1 = 2$)

Bounding Maximum Degree

Lemma-04

$size(x) \geq F_{k+2} \geq \Phi^k$ where $k = x.dergee$ for an node x in FH

Proof: Let s_k be minimum size of a node having degree k. (Bases cases are $s_0 = 1$, $s_1 = 2$)

$$size(x) \geq s_k$$

Bounding Maximum Degree

Lemma-04

$size(x) \geq F_{k+2} \geq \Phi^k$ where $k = x.dergee$ for an node x in FH

Proof: Let s_k be minimum size of a node having degree k. (Bases cases are $s_0 = 1$, $s_1 = 2$)

$$\begin{aligned} size(x) &\geq s_k \\ &\geq \; 2 + \sum_{i=2}^{k} s_{y_i.degree} \end{aligned}$$

Bounding Maximum Degree

Lemma-04

$size(x) \geq F_{k+2} \geq \Phi^k$ where $k = x.dergee$ for an node x in FH

Proof: Let s_k be minimum size of a node having degree k. (Bases cases are $s_0 = 1$, $s_1 = 2$)

$$\begin{aligned} size(x) &\geq s_k \\ &\geq \quad 2 + \sum_{i=2}^{k} s_{y_i.degree} \geq 2 + \sum_{i=2}^{k} s_{i-2} \end{aligned}$$

Bounding Maximum Degree

Lemma-04

$size(x) \geq F_{k+2} \geq \Phi^k$ where $k = x.dergee$ for an node x in FH

Proof: Let s_k be minimum size of a node having degree k. (Bases cases are $s_0 = 1$, $s_1 = 2$)

$$\begin{aligned} size(x) &\geq s_k \\ &\geq \quad 2 + \sum_{i=2}^{k} s_{y_i.degree} \geq 2 + \sum_{i=2}^{k} s_{i-2} \end{aligned}$$

Use induction to show $s_k \geq F_{k+2}$ (base case is trivial). Consider $i \geq 2$

Bounding Maximum Degree

Lemma-04

$size(x) \geq F_{k+2} \geq \Phi^k$ where $k = x.dergee$ for an node x in FH

Proof: Let s_k be minimum size of a node having degree k. (Bases cases are $s_0 = 1$, $s_1 = 2$)

$$\begin{aligned} size(x) &\geq s_k \\ &\geq 2 + \sum_{i=2}^{k} s_{y_i.degree} \geq 2 + \sum_{i=2}^{k} s_{i-2} \end{aligned}$$

Use induction to show $s_k \geq F_{k+2}$ (base case is trivial). Consider $i \geq 2$

$$s_k \geq 2 + \sum_{i=2}^{k} F_i$$

Bounding Maximum Degree

Lemma-04

$size(x) \geq F_{k+2} \geq \Phi^k$ where $k = x.dergee$ for an node x in FH

Proof: Let s_k be minimum size of a node having degree k. (Bases cases are $s_0 = 1$, $s_1 = 2$)

$$\begin{aligned} size(x) &\geq s_k \\ &\geq 2 + \sum_{i=2}^{k} s_{y_i.degree} \geq 2 + \sum_{i=2}^{k} s_{i-2} \end{aligned}$$

Use induction to show $s_k \geq F_{k+2}$ (base case is trivial). Consider $i \geq 2$

$$s_k \geq 2 + \sum_{i=2}^{k} F_i \geq 1 + \sum_{i=0}^{k} F_i$$

Bounding Maximum Degree

Lemma-04

$size(x) \geq F_{k+2} \geq \Phi^k$ where $k = x.dergee$ for an node x in FH

Proof: Let s_k be minimum size of a node having degree k. (Bases cases are $s_0 = 1$, $s_1 = 2$)

$$\begin{aligned} size(x) &\geq s_k \\ &\geq 2 + \sum_{i=2}^{k} s_{y_i.degree} \geq 2 + \sum_{i=2}^{k} s_{i-2} \end{aligned}$$

Use induction to show $s_k \geq F_{k+2}$ (base case is trivial). Consider $i \geq 2$

$$s_k \geq 2 + \sum_{i=2}^{k} F_i \geq 1 + \sum_{i=0}^{k} F_i = F_{k+2}$$

Bounding Maximum Degree

Lemma-04

$size(x) \geq F_{k+2} \geq \Phi^k$ where $k = x.dergee$ for an node x in FH

Proof: Let s_k be minimum size of a node having degree k. (Bases cases are $s_0 = 1$, $s_1 = 2$)

$$\begin{aligned} size(x) &\geq s_k \\ &\geq \quad 2 + \sum_{i=2}^{k} s_{y_i.degree} \geq 2 + \sum_{i=2}^{k} s_{i-2} \end{aligned}$$

Use induction to show $s_k \geq F_{k+2}$ (base case is trivial). Consider $i \geq 2$

$$\begin{aligned} s_k \quad &\geq \quad 2 + \sum_{i=2}^{k} F_i \geq 1 + \sum_{i=0}^{k} F_i = F_{k+2} \\ &\geq \quad \Phi^k \end{aligned}$$

Bounding Maximum Degree

Corollary

Maximum degree $D(n)$ of any node in an n-node FH is $O(\log n)$

Bounding Maximum Degree

Corollary

Maximum degree $D(n)$ of any node in an n-node FH is $O(\log n)$

Proof: Let x be a node in an n-node FH

Bounding Maximum Degree

Corollary

Maximum degree $D(n)$ of any node in an n-node FH is $O(\log n)$

Proof: Let x be a node in an n-node FH

- Let $k = x.degree$

Bounding Maximum Degree

Corollary

Maximum degree $D(n)$ of any node in an n-node FH is $O(\log n)$

Proof: Let x be a node in an n-node FH

- Let $k = x.degree$
- $n \geq size(x) \geq \Phi^k$

Bounding Maximum Degree

Corollary

Maximum degree $D(n)$ of any node in an n-node FH is $O(\log n)$

Proof: Let x be a node in an n-node FH

- Let $k = x.degree$
- $n \geq size(x) \geq \Phi^k$
- Takes log_Φ both side

$$k \leq \log_\Phi(n)$$

Bounding Maximum Degree

Corollary

Maximum degree $D(n)$ of any node in an n-node FH is $O(\log n)$

Proof: Let x be a node in an n-node FH

- Let $k = x.degree$
- $n \geq size(x) \geq \Phi^k$
- Takes log_Φ both side

$$k \leq \log_\Phi(n)$$

- Therefore, $D(n) \leq \log_\Phi(n)$

All-Pairs Shortest Paths (Floyd-Warshall)

- Floyd-Warshall algorithm considers the intermediate vertices of a shortest path

All-Pairs Shortest Paths (Floyd-Warshall)

- Floyd-Warshall algorithm considers the intermediate vertices of a shortest path

$$d_{ij}^{(k)} = \begin{cases} w_{ij} & \text{if } k = 0 \\ min(d_{ij}^{(k-1)}, d_{ik}^{(k)} + d_{kj}^{(k)}) & \text{otherwise} \end{cases}$$

All-Pairs Shortest Paths (Floyd-Warshall)

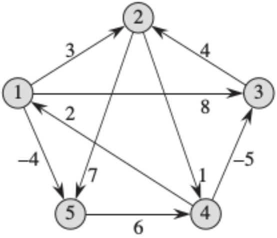
- Floyd-Warshall algorithm considers the intermediate vertices of a shortest path

$$d_{ij}^{(k)} = \begin{cases} w_{ij} & \text{if } k = 0 \\ min(d_{ij}^{(k-1)}, d_{ik}^{(k)} + d_{kj}^{(k)}) & \text{otherwise} \end{cases}$$

FLOYD-WARSHALL(W)

$n = W.rows$
$D^{(0)} = W$
for $k = 1$ **to** n
  let $D^{(k)} = (d_{ij}^{(k)})$ be a new $n \times n$ matrix
  for $i = 1$ **to** n
    for $j = 1$ **to** n
      $d_{ij}^{(k)} = \min\left(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right)$
return $D^{(n)}$

All-Pairs Shortest Paths



- Floyd-Warshall algorithm takes $\Theta(V^3)$ time

$$D^{(0)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \quad \Pi^{(0)} = \begin{pmatrix} \text{NIL} & 1 & 1 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & \text{NIL} & \text{NIL} \\ 4 & \text{NIL} & 4 & \text{NIL} & \text{NIL} \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(1)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \quad \Pi^{(1)} = \begin{pmatrix} \text{NIL} & 1 & 1 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & \text{NIL} & \text{NIL} \\ 4 & 1 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(2)} = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \quad \Pi^{(2)} = \begin{pmatrix} \text{NIL} & 1 & 1 & 2 & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & 2 & 2 \\ 4 & 1 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(3)} = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \quad \Pi^{(3)} = \begin{pmatrix} \text{NIL} & 1 & 1 & 2 & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & 2 & 2 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(4)} = \begin{pmatrix} 0 & 3 & -1 & 4 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix} \quad \Pi^{(4)} = \begin{pmatrix} \text{NIL} & 1 & 4 & 2 & 1 \\ 4 & \text{NIL} & 4 & 2 & 1 \\ 4 & 3 & \text{NIL} & 2 & 1 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ 4 & 3 & 4 & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(5)} = \begin{pmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix} \quad \Pi^{(5)} = \begin{pmatrix} \text{NIL} & 3 & 4 & 5 & 1 \\ 4 & \text{NIL} & 4 & 2 & 1 \\ 4 & 3 & \text{NIL} & 2 & 1 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ 4 & 3 & 4 & 5 & \text{NIL} \end{pmatrix}$$

Thank You!

Thank you very much for your attention! (Reference[1])

Queries ?

[1] [1] Book - *Introduction to Algorithm*, By THOMAS H. CORMEN, CHARLES E. LEISERSON, RONALD L. RIVEST, CLIFFORD STEIN