15.5
Algorithms via Basic Search

Algorithms via Basic Search - I

1. Given G and nodes u and v, can u reach v?
2. Given G and u, compute $\mathbf{rch}(u)$.

Use $\mathit{Explore}(G, u)$ to compute $\mathbf{rch}(u)$ in $O(n + m)$ time.

Algorithms via Basic Search - II

1. Given G and u, compute all v that can reach u, that is all v such that $u \in \text{rch}(v)$. Naive: $O(n(n+m))$

Definition (Reverse graph.)

Given $G = (V, E)$, G^{rev} is the graph with edge directions reversed $G^{rev} = (V, E')$ where $E' = \{(y, x) \mid (x, y) \in E\}$

Compute $\text{rch}(u)$ in G^{rev}!

1. **Correctness:** exercise
2. **Running time:** $O(n+m)$ to obtain G^{rev} from G and $O(n+m)$ time to compute $\text{rch}(u)$ via Basic Search. If both $Out(v)$ and $In(v)$ are available at each v then no need to explicitly compute G^{rev}. Can do $Explore(G, u)$ in G^{rev} implicitly.

Algorithms via Basic Search - II

1. Given G and u, compute all v that can reach u, that is all v such that $u \in \mathrm{rch}(v)$. Naive: $O(n(n+m))$

Definition (Reverse graph.)

Given $G = (V, E)$, G^{rev} is the graph with edge directions reversed $G^{rev} = (V, E')$ where $E' = \{(y, x) \mid (x, y) \in E\}$

Compute $\mathrm{rch}(u)$ in G^{rev}!

1. **Correctness:** exercise
2. **Running time:** $O(n+m)$ to obtain G^{rev} from G and $O(n+m)$ time to compute $\mathrm{rch}(u)$ via Basic Search. If both $Out(v)$ and $In(v)$ are available at each v then no need to explicitly compute G^{rev}. Can do $Explore(G, u)$ in G^{rev} implicitly.

Algorithms via Basic Search - II

1. Given G and u, compute all v that can reach u, that is all v such that $u \in \mathsf{rch}(v)$. Naive: $O(n(n+m))$

Definition (Reverse graph.)

Given $G = (V, E)$, G^{rev} is the graph with edge directions reversed $G^{rev} = (V, E')$ where $E' = \{(y, x) \mid (x, y) \in E\}$

Compute $\mathsf{rch}(u)$ in G^{rev}!

1. **Correctness:** exercise
2. **Running time:** $O(n+m)$ to obtain G^{rev} from G and $O(n+m)$ time to compute $\mathsf{rch}(u)$ via Basic Search. If both $Out(v)$ and $In(v)$ are available at each v then no need to explicitly compute G^{rev}. Can do $Explore(G, u)$ in G^{rev} implicitly.

Algorithms via Basic Search - II

1. Given G and u, compute all v that can reach u, that is all v such that $u \in \mathrm{rch}(v)$. Naive: $O(n(n+m))$

Definition (Reverse graph.)

Given $G = (V, E)$, G^{rev} is the graph with edge directions reversed $G^{rev} = (V, E')$ where $E' = \{(y, x) \mid (x, y) \in E\}$

Compute $\mathrm{rch}(u)$ in G^{rev}!

1. **Correctness:** exercise
2. **Running time:** $O(n + m)$ to obtain G^{rev} from G and $O(n + m)$ time to compute $\mathrm{rch}(u)$ via Basic Search. If both $Out(v)$ and $In(v)$ are available at each v then no need to explicitly compute G^{rev}. Can do $Explore(G, u)$ in G^{rev} implicitly.

Algorithms via Basic Search - III

$\mathrm{SCC}(G, u) = \{v \mid u \text{ is strongly connected to } v\}$

1. Find the strongly connected component containing node u. That is, compute $\mathrm{SCC}(G, u)$.

$\mathrm{SCC}(G, u) = \mathbf{rch}(G, u) \cap \mathbf{rch}(G^{rev}, u)$

Hence, $\mathrm{SCC}(G, u)$ can be computed with $Explore(G, u)$ and $Explore(G^{rev}, u)$. Total $O(n + m)$ time.

Why can $\mathbf{rch}(G, u) \cap \mathbf{rch}(G^{rev}, u)$ be done in $O(n)$ time?

Algorithms via Basic Search - III

$\mathrm{SCC}(G, u) = \{v \mid u \text{ is strongly connected to } v\}$

1. Find the strongly connected component containing node u. That is, compute $\mathrm{SCC}(G, u)$.

$\mathrm{SCC}(G, u) = \mathbf{rch}(G, u) \cap \mathbf{rch}(G^{rev}, u)$

Hence, $\mathrm{SCC}(G, u)$ can be computed with $Explore(G, u)$ and $Explore(G^{rev}, u)$. Total $O(n + m)$ time.

Why can $\mathbf{rch}(G, u) \cap \mathbf{rch}(G^{rev}, u)$ be done in $O(n)$ time?

Algorithms via Basic Search - III

$\mathrm{SCC}(G, u) = \{v \mid u \text{ is strongly connected to } v\}$

1. Find the strongly connected component containing node u. That is, compute $\mathrm{SCC}(G, u)$.

$\mathrm{SCC}(G, u) = \mathbf{rch}(G, u) \cap \mathbf{rch}(G^{rev}, u)$

Hence, $\mathrm{SCC}(G, u)$ can be computed with $Explore(G, u)$ and $Explore(G^{rev}, u)$. Total $O(n + m)$ time.

Why can $\mathbf{rch}(G, u) \cap \mathbf{rch}(G^{rev}, u)$ be done in $O(n)$ time?

Algorithms via Basic Search - III

$\mathrm{SCC}(G, u) = \{v \mid u \text{ is strongly connected to } v\}$

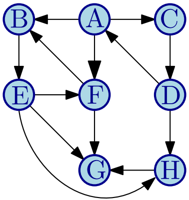
1. Find the strongly connected component containing node u. That is, compute $\mathrm{SCC}(G, u)$.

$\mathrm{SCC}(G, u) = \mathbf{rch}(G, u) \cap \mathbf{rch}(G^{rev}, u)$

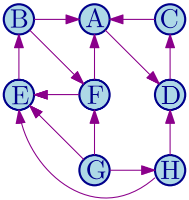
Hence, $\mathrm{SCC}(G, u)$ can be computed with $Explore(G, u)$ and $Explore(G^{rev}, u)$. Total $O(n + m)$ time.

Why can $\mathbf{rch}(G, u) \cap \mathbf{rch}(G^{rev}, u)$ be done in $O(n)$ time?

SCC I: Graph G and its reverse graph $G^{\mathbf{rev}}$



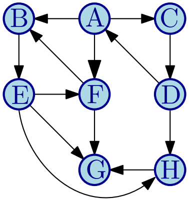
Graph G



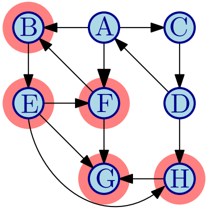
Reverse graph $G^{\mathbf{rev}}$

SCC II: Graph G a vertex F

.. and its reachable set **rch**(G, F)



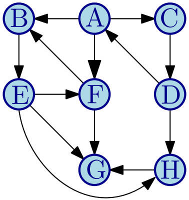
Graph G



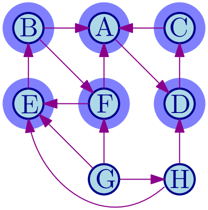
Reachable set of vertices from ***F***

SCC III: Graph G a vertex F

.. and the set of vertices that can reach it in G: **rch**(G^{rev}, F)



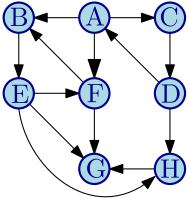
Graph G



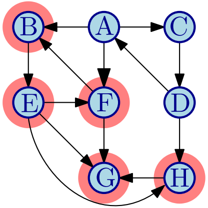
Set of vertices that can reach $\boldsymbol{F}$, computed via **DFS** in the reverse graph $\boldsymbol{G}^{\mathrm{rev}}$.

SCC IV: Graph G a vertex F and...

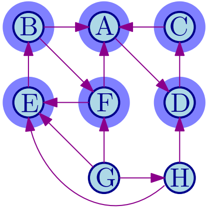
its strong connected component in G: $\mathrm{SCC}(\mathsf{G}, \mathsf{F})$



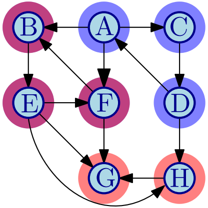
Graph G



rch(G, ***F***)



rch($\mathsf{G}^{\mathrm{rev}}$, ***F***)



$$\mathrm{SCC}(\mathsf{G}, \boldsymbol{F}) = \mathbf{rch}(\mathsf{G}, \boldsymbol{F}) \cap \mathbf{rch}(\mathsf{G}^{\mathrm{rev}}, \boldsymbol{F})$$

Algorithms via Basic Search - IV

1. Is G strongly connected?

Pick arbitrary vertex u. Check if $\mathrm{SCC}(G, u) = V$.

Algorithms via Basic Search - IV

1. Is G strongly connected?

Pick arbitrary vertex u. Check if $\mathrm{SCC}(G, u) = V$.

Algorithms via Basic Search - V

1. Find <u>all</u> strongly connected components of G.

```
While G is not empty do
    Pick arbitrary node u
    find S = SCC(G, u)
    Remove S from G
```

Question: Why doesn't removing one strong connected components affect the other strong connected components?

Running time: $O(n(n + m))$.

Question: Can we do it in $O(n + m)$ time?

Algorithms via Basic Search - V

1. Find all strongly connected components of G.

```
While G is not empty do
    Pick arbitrary node u
    find S = SCC(G, u)
    Remove S from G
```

Question: Why doesn't removing one strong connected components affect the other strong connected components?

Running time: $O(n(n+m))$.

Question: Can we do it in $O(n+m)$ time?

Algorithms via Basic Search - V

1. Find all strongly connected components of G.

```
While G is not empty do
    Pick arbitrary node u
    find S = SCC(G, u)
    Remove S from G
```

Question: Why doesn't removing one strong connected components affect the other strong connected components?

Running time: $O(n(n+m))$.

Question: Can we do it in $O(n+m)$ time?

Algorithms via Basic Search - V

1. Find all strongly connected components of G.

```
While G is not empty do
    Pick arbitrary node u
    find S = SCC(G, u)
    Remove S from G
```

Question: Why doesn't removing one strong connected components affect the other strong connected components?

Running time: $O(n(n+m))$.

Question: Can we do it in $O(n+m)$ time?

Algorithms via Basic Search - V

1. Find all strongly connected components of G.

```
While G is not empty do
    Pick arbitrary node u
    find S = SCC(G, u)
    Remove S from G
```

Question: Why doesn't removing one strong connected components affect the other strong connected components?

Running time: $O(n(n+m))$.

Question: Can we do it in $O(n+m)$ time?

THE END

...

(for now)