

Advanced Core in Algorithm Design #2

算法設計要論 第2回

Yasushi Kawase
河瀬 康志

October 12th, 2021

last update: 12:07pm, November 6, 2021

Lec. #	Date	Topics
1	10/5	Introduction, Stable matching
2	10/12	Basics of Algorithm Analysis, Graphs
3	10/19	Greedy Algorithms (1/2)
4	10/26	Greedy Algorithms (2/2)
5	11/2	Divide and Conquer (1/2)
6	11/9	Divide and Conquer (2/2)
7	11/16	Dynamic Programming (1/2)
8	11/30	Dynamic Programming (2/2)
9	12/7	Network Flow (1/2)
10	12/14	Network Flow (2/2)
11	12/21	NP and Computational Intractability
12	1/4	Approximation Algorithms (1/2)
13	1/11	Approximation Algorithms (2/2)
14	1/18	Final Examination

- 1 Asymptotic Order of Growth
- 2 Basic Algorithms
- 3 Graph Traversal

Relationship between Input Size and Running Time

	\sqrt{n}	n	$n \log n$	n^2	2^n	$n!$
1 sec.	$1.0 \cdot 10^{16}$	$1.0 \cdot 10^8$	$6.4 \cdot 10^6$	10000	26	11
1 min.	$3.6 \cdot 10^{19}$	$6.0 \cdot 10^9$	$3.1 \cdot 10^8$	77500	32	12
1 hour	$1.3 \cdot 10^{23}$	$3.6 \cdot 10^{11}$	$1.5 \cdot 10^{10}$	600000	38	15
1 day	$7.5 \cdot 10^{25}$	$8.6 \cdot 10^{12}$	$3.3 \cdot 10^{11}$	$2.9 \cdot 10^6$	42	16
1 month	$6.7 \cdot 10^{28}$	$2.6 \cdot 10^{14}$	$8.7 \cdot 10^{12}$	$1.6 \cdot 10^7$	47	17
1 year	$9.7 \cdot 10^{30}$	$3.1 \cdot 10^{15}$	$9.7 \cdot 10^{13}$	$5.6 \cdot 10^7$	51	18
1 century	$9.7 \cdot 10^{34}$	$3.1 \cdot 10^{17}$	$8.5 \cdot 10^{15}$	$5.6 \cdot 10^8$	58	20

- Maximum sizes that can be calculated in limited times
- assumption: 10^8 calculations per second

Asymptotic Order

notations to represent order for sufficiently large n

$$1 < \log n < \sqrt{n} < n < n \log n < n^2 < n^{100} < 2^n < 3^n < n! < n^n$$

Asymptotic upper bound: $f(n) = O(g(n))$

$$\exists k > 0, \exists n_0, \forall n > n_0, f(n) \leq k \cdot g(n)$$

Asymptotic lower bound: $f(n) = \Omega(g(n))$

$$\exists k > 0, \exists n_0, \forall n > n_0, f(n) \geq k \cdot g(n)$$

Asymptotically tight bound: $f(n) = \Theta(g(n))$

$$f(n) = O(g(n)) \text{ and } f(n) = \Omega(g(n))$$

Examples

- $1000n = \Theta(n)$
- $18n^2 + 5n + 1 = \Theta(n^2)$
- $10(n + 5)^8 = \Theta(n^8)$
- $\log_{10} n = \frac{\log_2 n}{\log_2 10} = \Theta(\log n)$ (base does not matter)
- $n = O(2^n)$ (big-O is just an upper bound)
- $5 \cdot 2^{n+3} = \Theta(2^n)$
- $\log(n!) = \Theta(n \log n)$ ($\because (n/2)^{n/2} \leq n! \leq n^n$)
- $n^{100} = O(1.1^n)$ (see next)
- $(\log n)^{100} = O(n^{0.001})$ (see after next)

Proposition

For every $r > 1$ and $d > 0$, we have $n^d = O(r^n)$

Proof For $n \geq d$, we have

$$\begin{aligned} r^n &= (1 + (r - 1))^n \\ &= \sum_{k=0}^n \binom{n}{k} (r - 1)^k \\ &\geq \binom{n}{d} (r - 1)^d = \frac{n}{d} \cdot \frac{n-1}{d-1} \cdot \dots \cdot \frac{n-d+1}{1} \cdot (r - 1)^d \\ &\geq \frac{(r - 1)^d}{\underbrace{d^d}_{\text{constant}}} \cdot n^d \end{aligned}$$

Proposition

For every $b > 1$, $d > 0$ and $\epsilon > 0$, we have $(\log_b n)^d = O(n^\epsilon)$

Proof By setting $m := \log_b n$, we have

$$n^\epsilon = (b^m)^\epsilon = (b^\epsilon)^m$$

As $m^d = O((b^\epsilon)^m)$,

$$(\log_b n)^d = O(n^\epsilon)$$

Transitivity

Proposition

If $f(n) = O(g(n))$ and $g(n) = O(h(n))$, then $f(n) = O(h(n))$

$$\begin{cases} \exists k, \exists n_0, \forall n > n_0, f(n) \leq k \cdot g(n) \\ \exists k', \exists n'_0, \forall n > n'_0, g(n) \leq k' \cdot h(n) \end{cases} \quad \longrightarrow \quad \forall n > \max\{n_0, n'_0\}, f(n) \leq k \cdot k' \cdot h(n)$$

Proposition

If $f(n) = \Omega(g(n))$ and $g(n) = \Omega(h(n))$, then $f(n) = \Omega(h(n))$

$$\begin{cases} \exists k, \exists n_0, \forall n > n_0, f(n) \geq k \cdot g(n) \\ \exists k', \exists n'_0, \forall n > n'_0, g(n) \geq k' \cdot h(n) \end{cases} \quad \longrightarrow \quad \forall n > \max\{n_0, n'_0\}, f(n) \geq k \cdot k' \cdot h(n)$$

Proposition

If $f(n) = \Theta(g(n))$ and $g(n) = \Theta(h(n))$, then $f(n) = \Theta(h(n))$

Find the smallest one.

1. $\Theta(e^n)$
2. $\Theta(n!)$
3. $\Theta(n^{\log_e n})$
4. $\Theta((\log_e n)^n)$

Outline

- 1 Asymptotic Order of Growth
- 2 Basic Algorithms
- 3 Graph Traversal

Problem

given n numbers a_1, a_2, \dots, a_n , compute the maximum value

Linear Search

Problem

given n numbers a_1, a_2, \dots, a_n , compute the maximum value

Algorithm

```
 $x \leftarrow a_1;$   
for  $i \leftarrow 2, 3, \dots, n$  do  
   $x \leftarrow \max\{x, a_i\};$   
Return  $x;$ 
```


Linear Search

Problem

given n numbers a_1, a_2, \dots, a_n , compute the maximum value

Algorithm

```
 $x \leftarrow a_1;$   
for  $i \leftarrow 2, 3, \dots, n$  do  
   $x \leftarrow \max\{x, a_i\};$   
Return  $x;$ 
```

→ $O(n)$ time

Binary Search

Problem

given a_1, a_2, \dots, a_n with $a_1 < a_2 < \dots < a_n$ and x , find index of x

Binary Search

Problem

given a_1, a_2, \dots, a_n with $a_1 < a_2 < \dots < a_n$ and x , find index of x

Algorithm

Let $\ell \leftarrow 1$ and $h \leftarrow n + 1$;

while $\ell < h$ **do**

```
     $m \leftarrow \lfloor (\ell + h) / 2 \rfloor$ ;  
    if  $x < a_m$  then  $h \leftarrow m$ ;  
    else if  $x > a_m$  then  $\ell \leftarrow m + 1$ ;  
    else Return  $m$ ;
```

Return No such index;

Binary Search

Problem

given a_1, a_2, \dots, a_n with $a_1 < a_2 < \dots < a_n$ and x , find index of x

Algorithm

Let $\ell \leftarrow 1$ and $h \leftarrow n + 1$;

while $\ell < h$ **do**

```
     $m \leftarrow \lfloor (\ell + h) / 2 \rfloor$ ;  
    if  $x < a_m$  then  $h \leftarrow m$ ;  
    else if  $x > a_m$  then  $\ell \leftarrow m + 1$ ;  
    else Return  $m$ ;
```

Return No such index;

→ $O(\log n)$ time

Problem

given a_1, a_2, \dots, a_n and x , find $I \subseteq \{1, \dots, n\}$ such that $\sum_{i \in I} a_i = x$

Brute Force

Problem

given a_1, a_2, \dots, a_n and x , find $I \subseteq \{1, \dots, n\}$ such that $\sum_{i \in I} a_i = x$

Algorithm

```
foreach subset  $I$  of  $\{1, 2, \dots, n\}$  do  
  if  $\sum_{i \in I} a_i = x$  then Return  $I$ ;  
Return No such  $I$ ;
```


Brute Force

Problem

given a_1, a_2, \dots, a_n and x , find $I \subseteq \{1, \dots, n\}$ such that $\sum_{i \in I} a_i = x$

Algorithm

```
foreach subset  $I$  of  $\{1, 2, \dots, n\}$  do  
  if  $\sum_{i \in I} a_i = x$  then Return  $I$ ;  
Return No such  $I$ ;
```

→ $O(n \cdot 2^n)$ time

Outline

- 1 Asymptotic Order of Growth
- 2 Basic Algorithms
- 3 Graph Traversal**

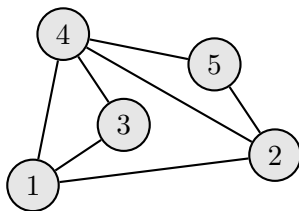
Undirected Graph $G = (V, E)$

- V : set of vertices. usually $n := |V|$
- E : set of **edges**. usually $m := |E|$

two-element subsets of V

Example:

- $V = \{1, 2, 3, 4, 5\}$
- $E = \{\{1, 2\}, \{1, 3\}, \{1, 4\}, \{2, 4\}, \{2, 5\}, \{3, 4\}, \{4, 5\}\}$



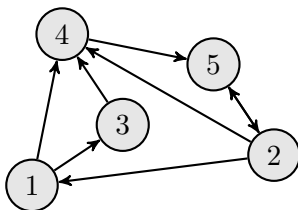
Directed Graph $G = (V, E)$

- V : set of vertices. usually $n := |V|$
- E : set of **edges**. usually $m := |E|$

pairs of vertices

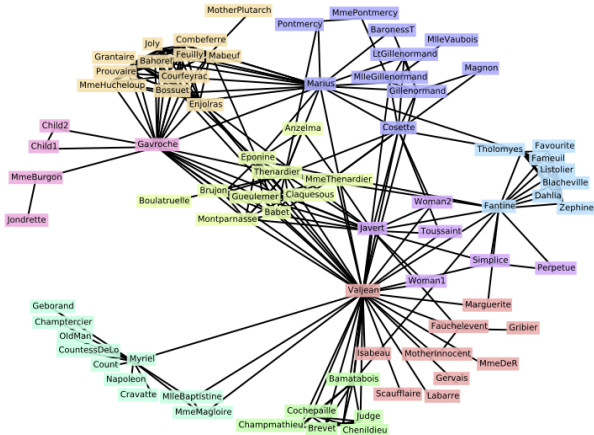
Example:

- $V = \{1, 2, 3, 4, 5\}$
- $E = \{(1, 3), (1, 4), (2, 1), (2, 4), (2, 5), (3, 4), (4, 5), (5, 2)\}$



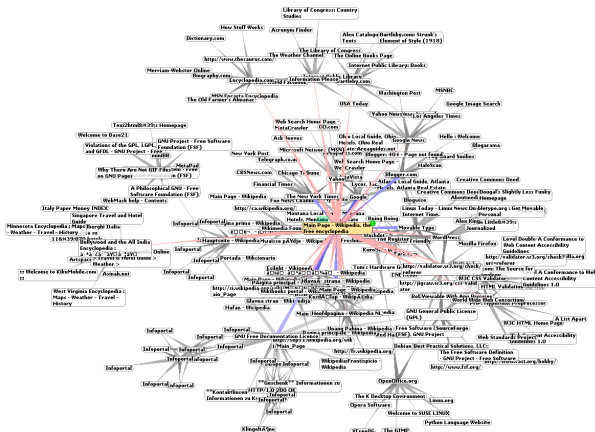
Example (2/3): Les Misérables

the interactions between characters ($n = 77$, $m = 254$)
created by Donald Knuth



Example (3/3): Webgraph

directed links between pages of the World Wide Web



https://en.wikipedia.org/wiki/Graph_drawing#/media/File:WorldWideWebAroundWikipedia.png

Adjacency matrix

- ★ matrix $A \in \{0, 1\}^{n \times n}$ where $a_{ij} = 1$ iff $(v_i, v_j) \in E$
- space complexity: $O(n^2)$
- check $(v_i, v_j) \in E$: $O(1)$
- suitable for dense graph

Adjacency list

- ★ n linked list, each describes the set of neighbors
- space complexity: $O(n + m)$
- check $(v_i, v_j) \in E$: $O(d)$
- suitable for sparse graph

d : maximum degree

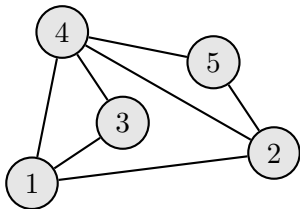
Representing Graphs — Example

Adjacency matrix

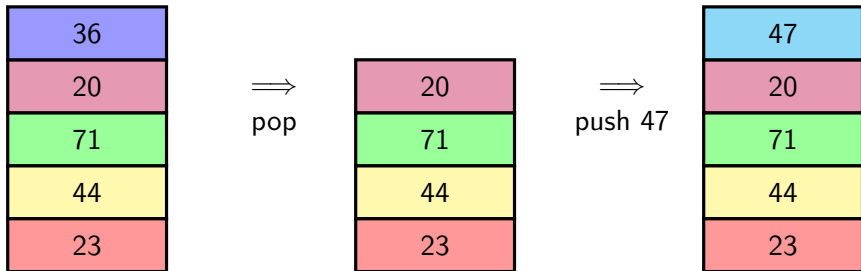
$$\begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

Adjacency list

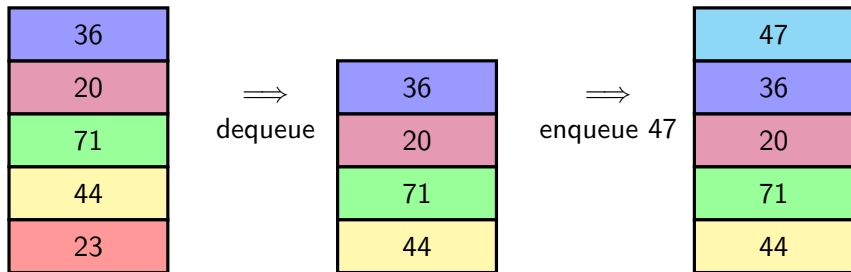
- 1 [2,3,4]
- 2 [1,4,5]
- 3 [1,4]
- 4 [1,2,3,5]
- 5 [2,4]



- a data structure that follows the LIFO principle
- two basic operations
 - push: add an element to the collection ($O(1)$ time)
 - pop: remove the most recently added element ($O(1)$ time)



- a data structure that follows the FIFO principle
- two basic operations
 - enqueue: add an element to the collection ($O(1)$ time)
 - dequeue: remove the earliest added element ($O(1)$ time)



Depth First Search

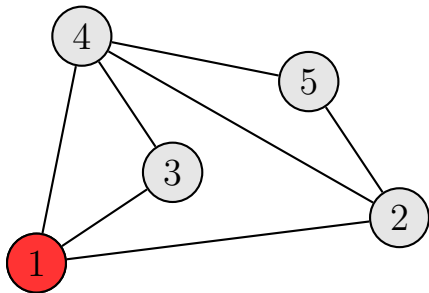
- an algorithm for graph traversal (determining s - t connectivity)
- push the next candidate to be searched on a **stack**
- $O(m + n)$ time (if the graph is represented by adjacency list)

DFS algorithm starting from $s \in V$

```
explored[u]  $\leftarrow$  False ( $\forall u \in V$ ) and let  $S$  be a stack only with  $s$ ;  
while  $S$  is not empty do  
    pop  $S$  and let  $u$  be the popped vertex;  
    if explored[ $u$ ] then continue;  
    explored[ $u$ ]  $\leftarrow$  True;  
    foreach  $\{u, v\} \in E$  incident to  $u$  do  
        if explored[ $v$ ] then push  $v$  to  $S$ ;
```

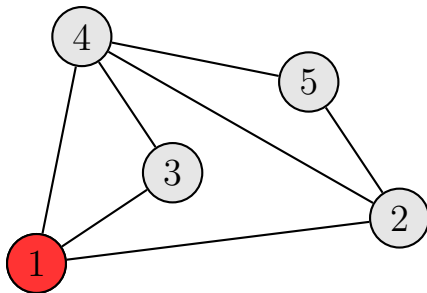

Example of DFS

- $\text{stack} = []$
- $u = 1$
-



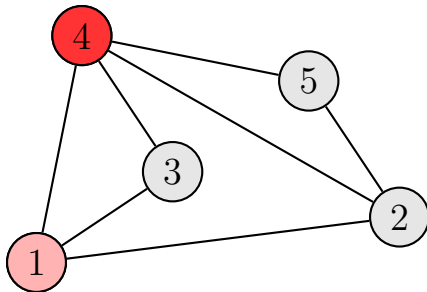
Example of DFS

- $\text{stack} = [2, 3, 4]$
- $u = 1$
- push 2, 3, 4



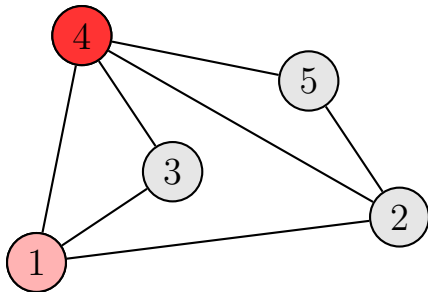
Example of DFS

- $\text{stack} = [2, 3]$
- $u = 4$
- pop



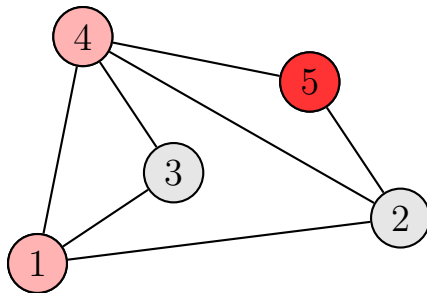
Example of DFS

- $\text{stack} = [2, 3, 2, 3, 5]$
- $u = 4$
- push 2,3,5



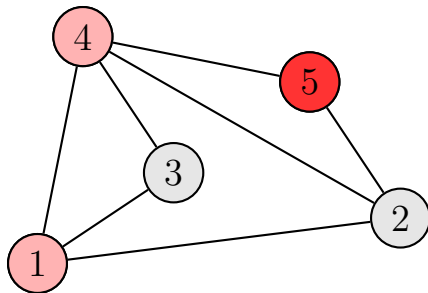
Example of DFS

- $\text{stack} = [2, 3, 2, 3]$
- $u = 5$
- pop



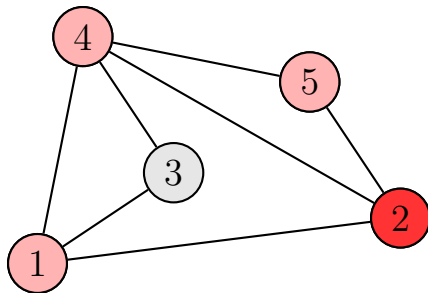
Example of DFS

- $\text{stack} = [2, 3, 2, 3, 2]$
- $u = 5$
- push 2



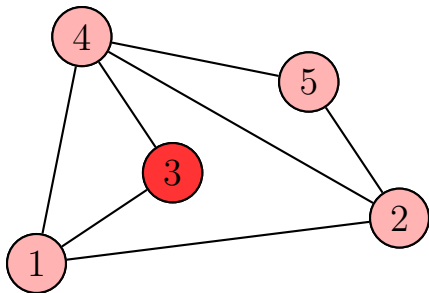
Example of DFS

- $\text{stack} = [2, 3, 2, 3]$
- $u = 2$
- pop



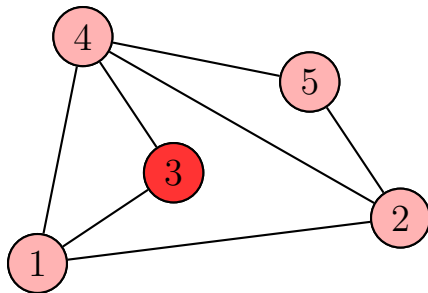
Example of DFS

- $\text{stack} = [2, 3, 2]$
- $u = 3$
- pop



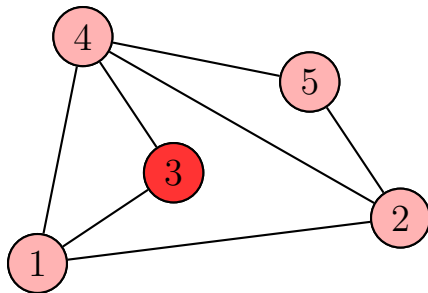
Example of DFS

- $\text{stack} = [2, 3]$
- $u = 3$
- pop



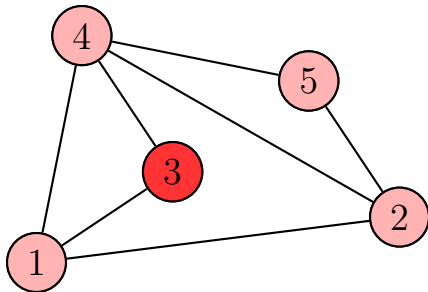
Example of DFS

- $\text{stack}=[2]$
- $u = 3$
- pop



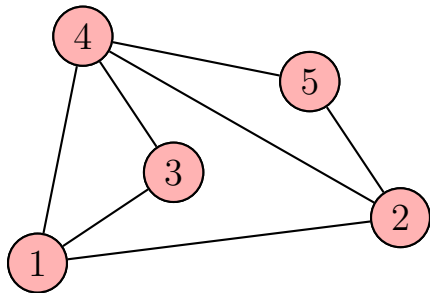
Example of DFS

- $\text{stack} = []$
- $u = 3$
- pop



Example of DFS

- $\text{stack} = []$
- $u = 3$
-



Breadth First Search

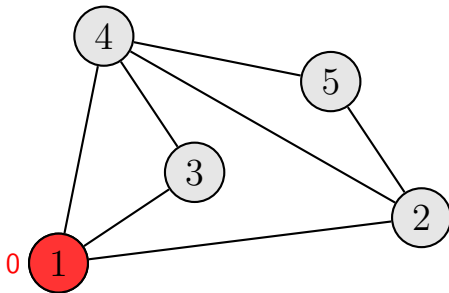
- an algorithm for graph traversal (determining s - t connectivity and dist.)
- enqueue the next candidate to be searched on a **queue**
- $O(m + n)$ time (if the graph is represented by adjacency list)

BFS algorithm starting from $s \in V$

```
dist[u]  $\leftarrow \infty$  ( $\forall u \in V$ ) and let  $Q$  be a queue only with  $(s, 0)$ ;  
while  $Q$  is not empty do  
    dequeue  $Q$  and let  $(u, d)$  be the dequeued vertex-distance pair;  
    if dist[u] <  $\infty$  then continue;  
    dist[u]  $\leftarrow d$ ;  
    foreach  $\{u, v\} \in E$  incident to  $u$  do  
        if explored[v] then enqueue  $(v, d + 1)$  to  $Q$ ;
```

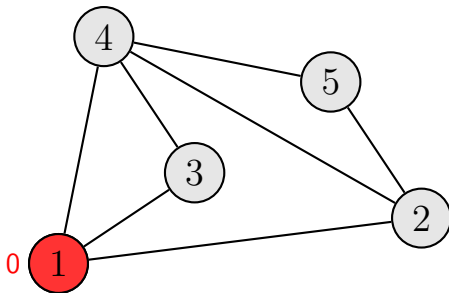

Example of BFS

- `queue=[]`
- $(u, d) = (1, 0)$
-



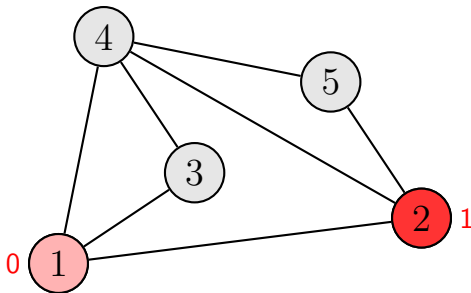
Example of BFS

- $\text{queue} = [(2, 1), (3, 1), (4, 1)]$
- $(u, d) = (1, 0)$
- enqueue 2, 3, 4



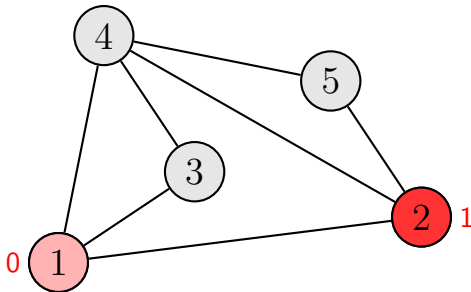
Example of BFS

- $\text{queue} = [(3, 1), (4, 1)]$
- $(u, d) = (2, 1)$
- dequeue



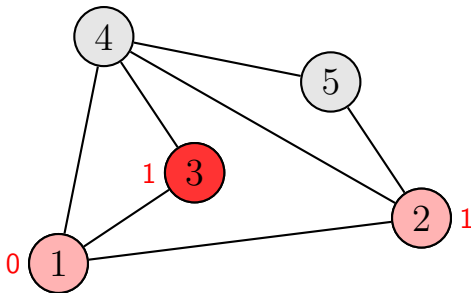
Example of BFS

- $\text{queue} = [(3, 1), (4, 1), (4, 2), (5, 2)]$
- $(u, d) = (2, 1)$
- enqueue 4, 5



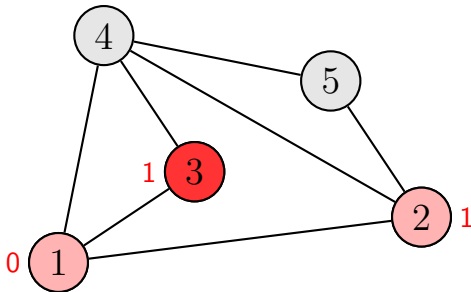
Example of BFS

- $\text{queue} = [(4, 1), (4, 2), (5, 2)]$
- $(u, d) = (3, 1)$
- dequeue



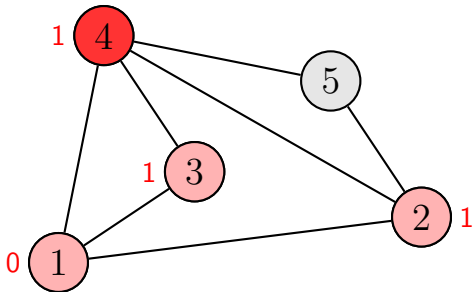
Example of BFS

- $\text{queue} = [(4, 1), (4, 2), (5, 2), (4, 2)]$
- $(u, d) = (3, 1)$
- enqueue 4



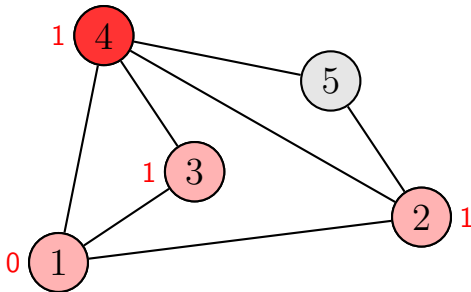
Example of BFS

- $\text{queue} = [(4, 2), (5, 2), (4, 2)]$
- $(u, d) = (4, 1)$
- dequeue



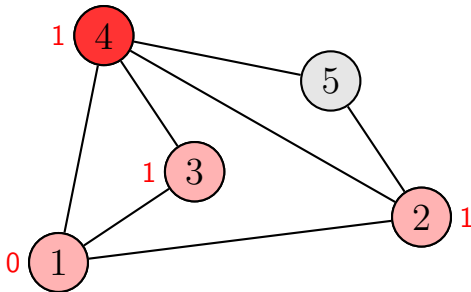
Example of BFS

- $\text{queue} = [(4, 2), (5, 2), (4, 2), (5, 2)]$
- $(u, d) = (4, 1)$
- enqueue 5



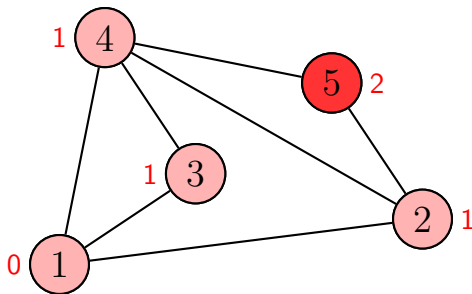
Example of BFS

- $\text{queue} = [(5, 2), (4, 2), (5, 2)]$
- $(u, d) = (4, 1)$
- dequeue



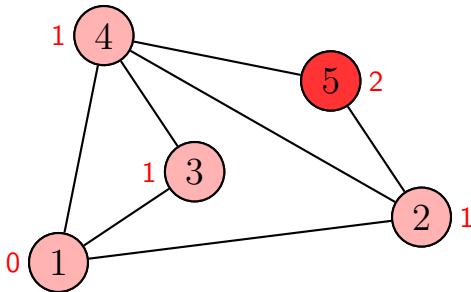
Example of BFS

- $\text{queue} = [(4, 2), (5, 2)]$
- $(u, d) = (5, 2)$
- dequeue



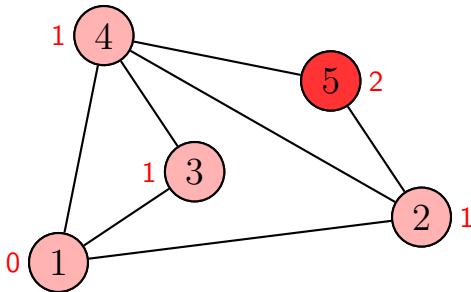
Example of BFS

- $\text{queue} = [(5, 2)]$
- $(u, d) = (5, 2)$
- dequeue



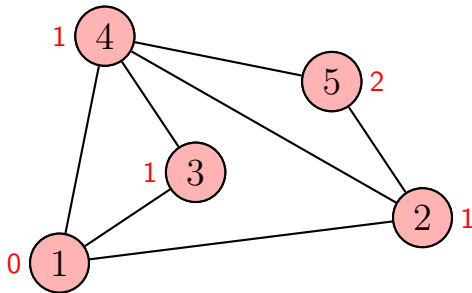
Example of BFS

- `queue=[]`
- $(u, d) = (5, 2)$
- `dequeue`



Example of BFS

- `queue=[]`
- $(u, d) = (5, 2)$
-



Depth First Search

- implemented using Stack
- may not give the shortest paths
- require less memory

Breadth First Search

- implemented using Queue
- give the shortest paths
- require more memory

