Metody szukania na grafach

dr hab. inż. Jerzy Balicki, prof. nadzw.

Dlaczego szukamy?

1. Dlaczego próbujemy czekolady z rybami i śpimy w lodowych łożach?
2. Kto smakował sorbet o smaku tequili i musztardy, a kto próbował trufle czekoladowe z octem winnym i anchois?
3. Wielu wybiera takie sposoby spędzania wakacji, wolnego czasu czy imprezy, o których z góry wiadomo, że nie będą zbyt przyjemne.
4. Pojedynki na jedzenie i picie, emeryci biorący udział w przeróżnych konferencjach, sporty ekstremalne, rekordy w księdze Guinessa …
5. Ale także zwiedzający ZOO, pasażerowie lubiący podróże pociągami i samolotami oraz osoby, których celem jest zwiedzenie wszystkich 50 stanów USA lub państw Ameryki Południowej. Nawigacja po Internecie …
6. A co ze studentami i wykładowcami? Dlaczego i czego szukamy?

Dlaczego szukamy?

- **Ponieważ mamy okazję do gromadzenia nowych doświadczeń i budowy czegoś w rodzaju "CV opartego na doświadczeniach” (teza potwierdzona empirycznie)**
- **Chcemy jak najwydajniej spędzać czas. Jak najlepiej i najpełniej przeżyć życie.**

Metody szukania na grafach

1. szukanie **wszerz** (ang. ***breadth first search***)
2. szukanie **w głąb** (ang. ***depth first search***)
3. metoda równomiernych kosztów/najkrótszej ścieżki (metoda **Dijkstry**)
4. metody heurystyczne – algorytm **A***
5. algorytmy **mrówkowe**
6. metody szukania na grafach **AND/OR**
7. metoda **minimax**
8. metoda **$\alpha - \beta$**

Przeszukiwanie grafu

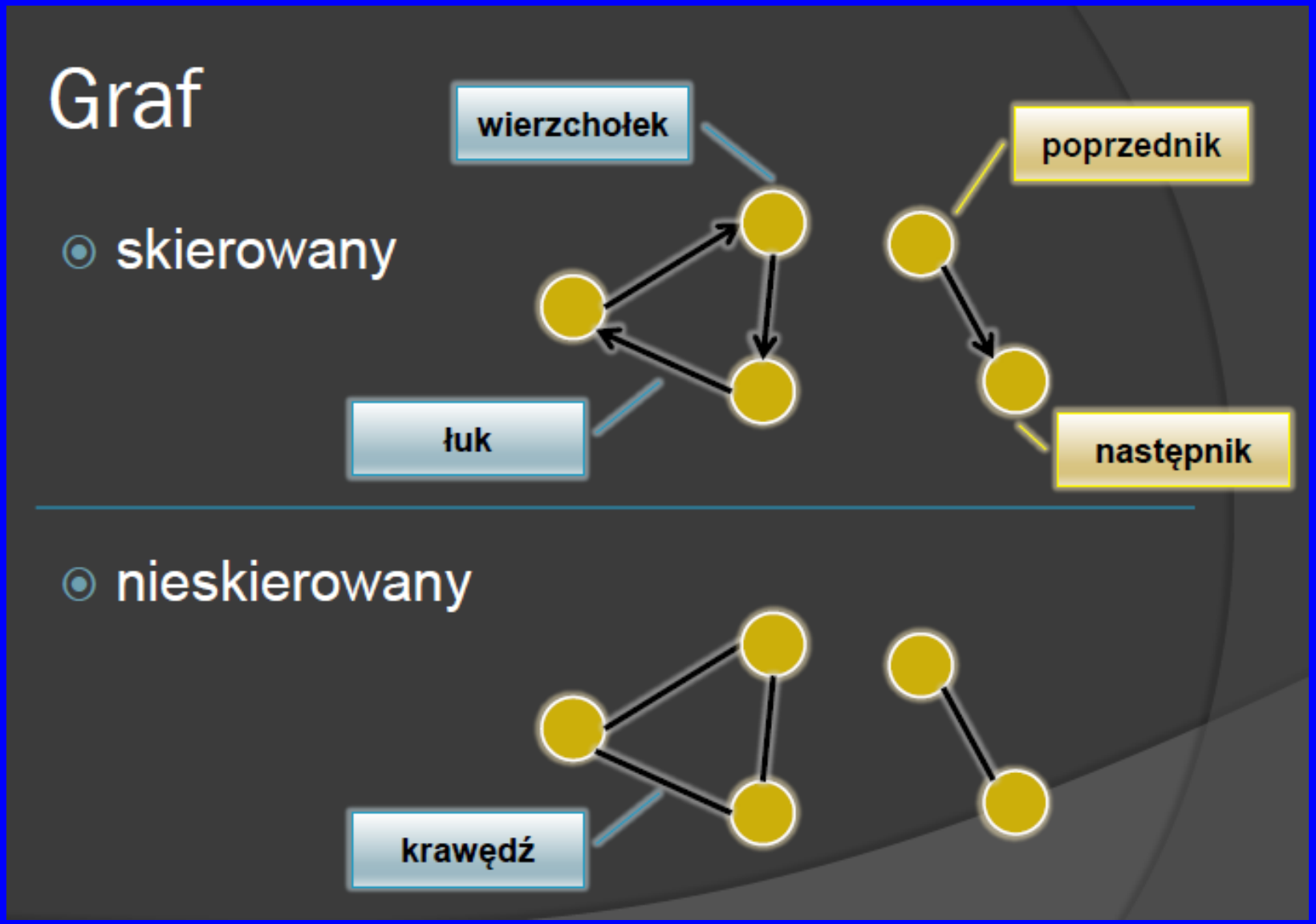
Czynność polegająca na *odwiedzeniu* w pewien usystematyzowany sposób *wszystkich wierzchołków grafu* w celu zebrania potrzebnych informacji (np. rekonesans przed atakiem hakerskim, analiza węzłów w SK)

Często podczas przejścia grafu rozwiązujemy już jakiś problem, ale przeważnie jest to tylko wstęp do wykonania właściwego algorytmu.

Stosuje się dwie podstawowe metody przeszukiwania grafów:

- **przeszukiwanie wszerz (BFS)**
- **przeszukiwanie w głąb (DFS)**

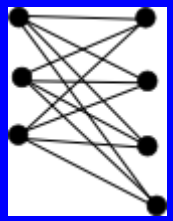
Grafy skierowane i nieskierowane



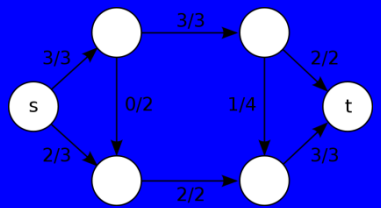
Szukanie wszerz

W problemach z teorii grafów:

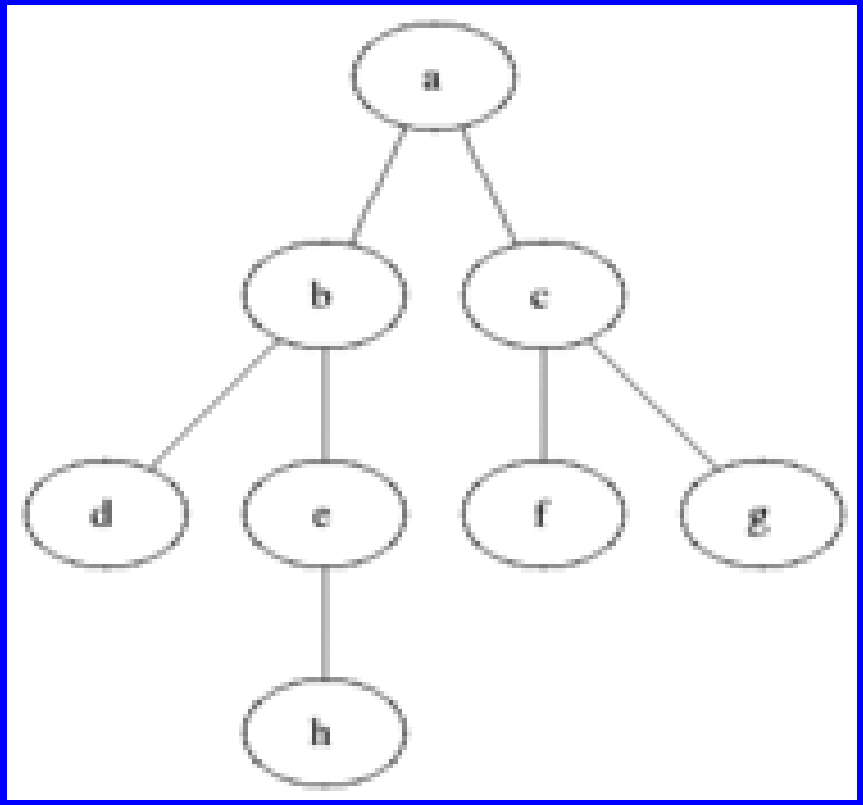
- **Odnalezienie wszystkich połączonych węzłów – gdzie znajduje się rozwiązanie gry?**
- Wyznaczenie najkrótszej ścieżki między dwoma wierzchołkami (*nie uwzględnia się wag krawędzi w grafie ważonym*) - wygrać partię szachów w jak najmniejszej liczbie ruchów.

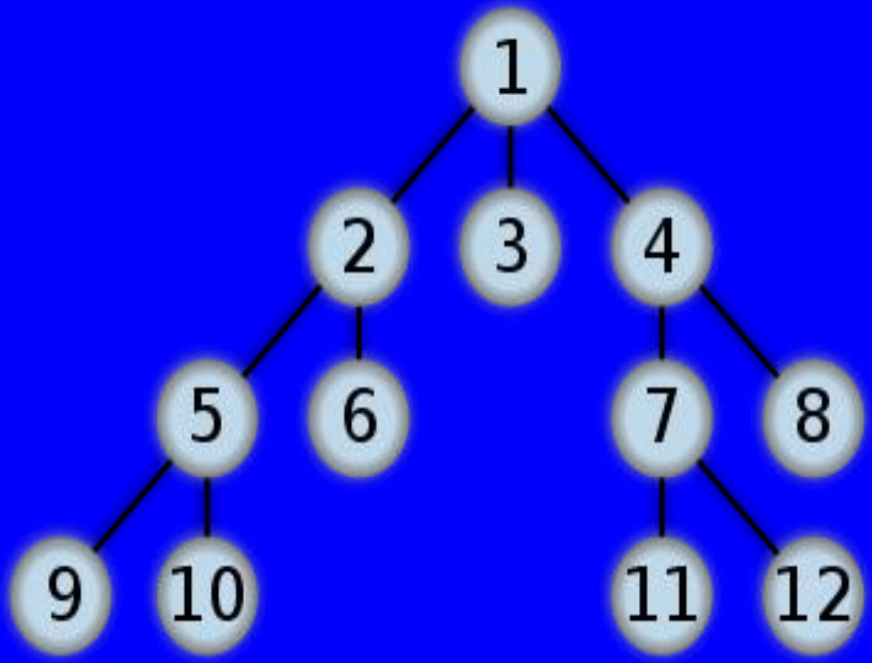


- **Sprawdzenie czy graf jest dwudzielny**
- Wyznaczenie maksymalnego przepływu w sieci – projektowanie struktury topologicznej WAN.

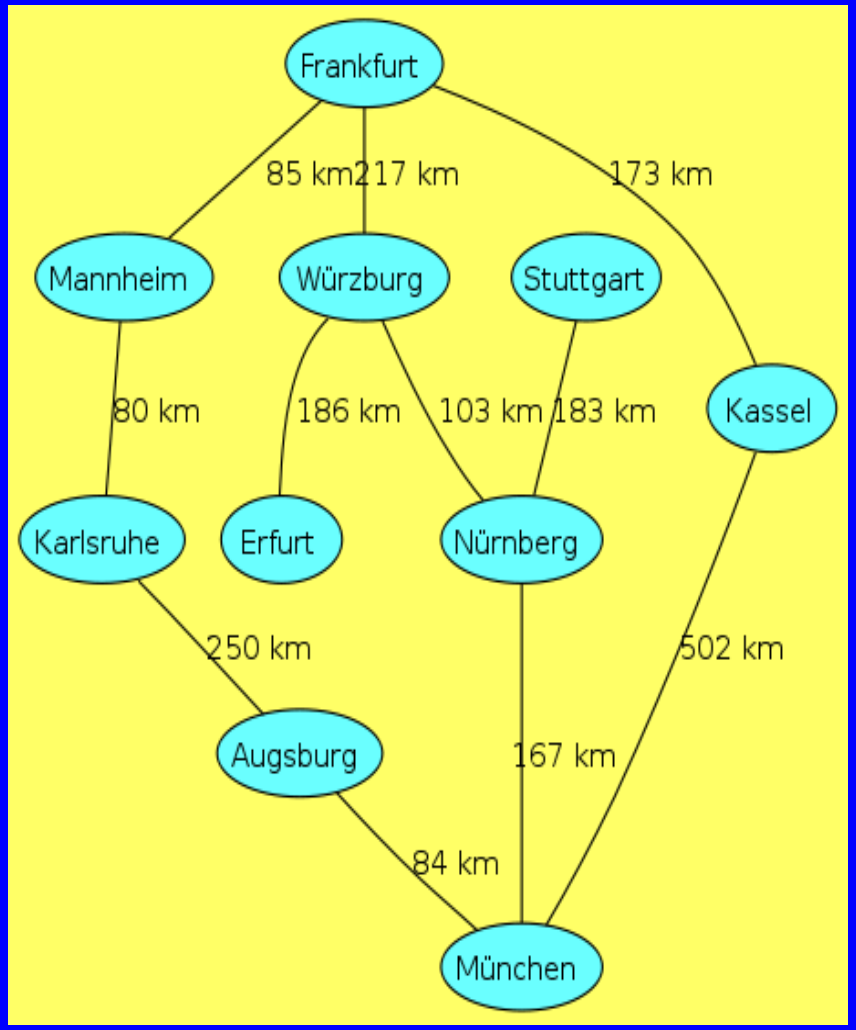


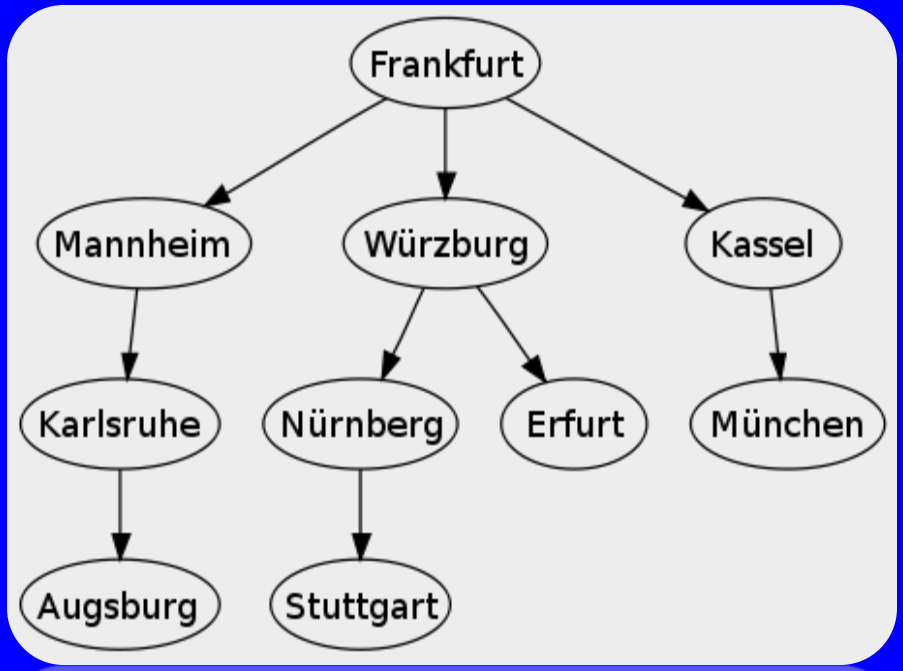
Szukanie wszerz

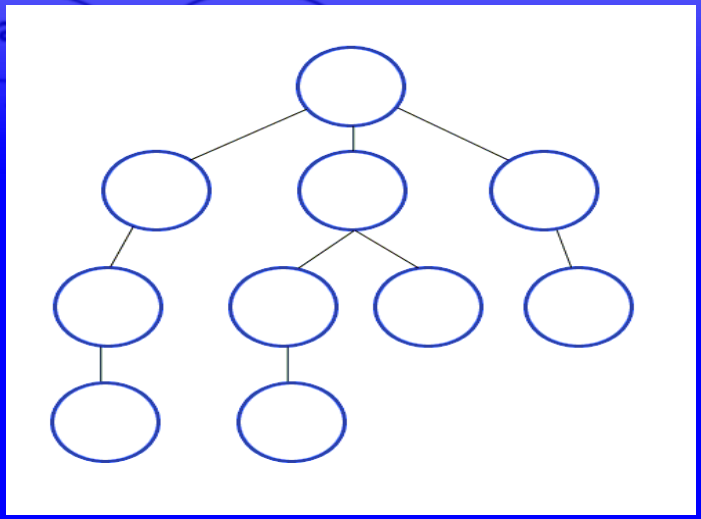




Szukanie wszerz na mapie





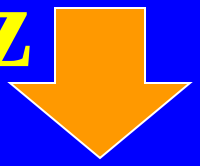


Ustawianie sekwencji ośmiu cyfr

2	8	3
1	6	4
7		5

Stan początkowy (losowy)

Szukanie wszerz



1	2	3
8		4
7	6	5

Cel gry

Ustawianie sekwencji ośmiu cyfr

1

2	8	3
1	6	4
7		5

Stan początkowy (losowy)

Dopuszczalne ruchy

2

2	8	3
1	6	4
	7	5

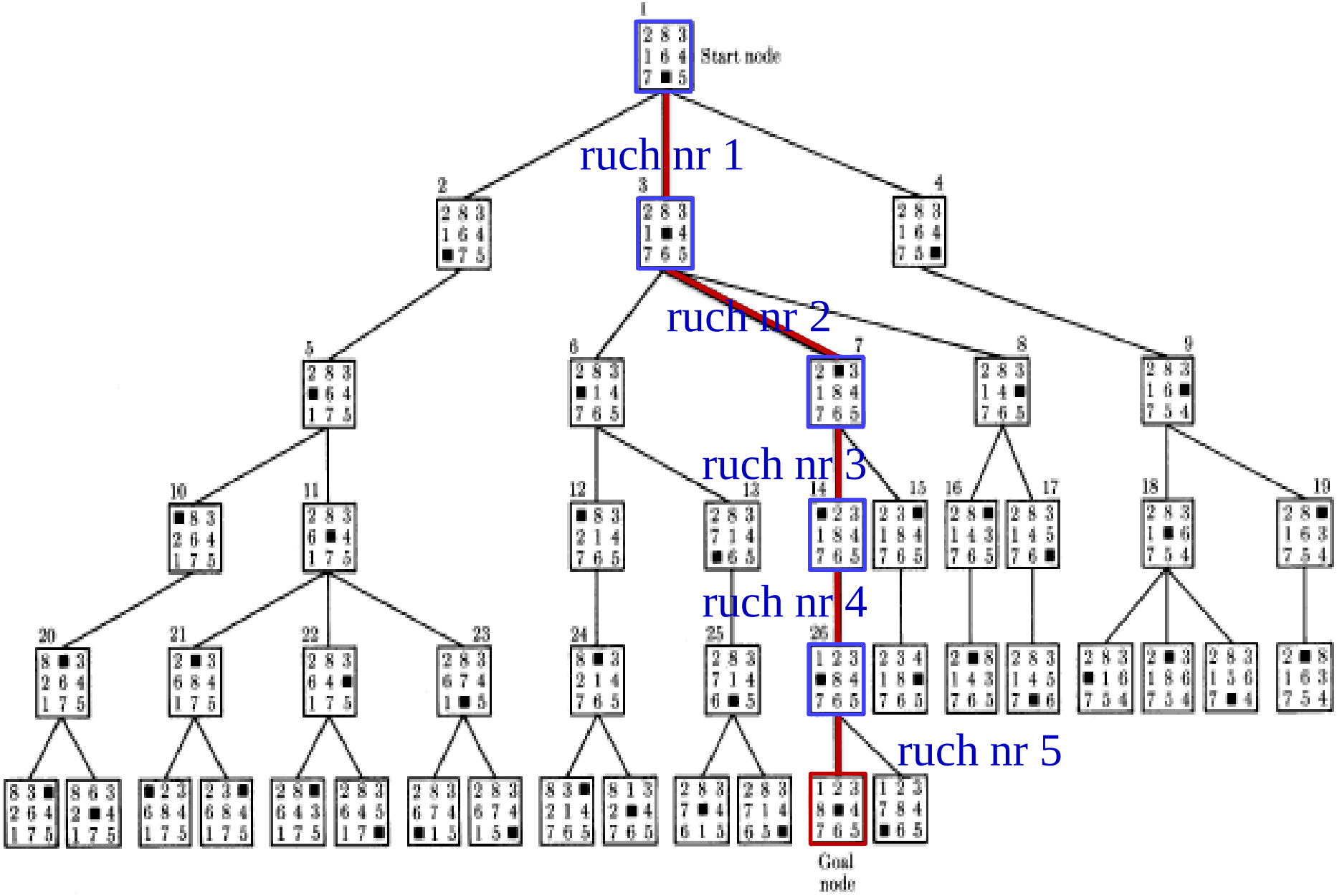
3

2	8	3
1		4
7	6	5

4

2	8	3
1	6	4
7	5	

Graf stanów gry dla zadanego korzenia



Algorytm szukania wszerz

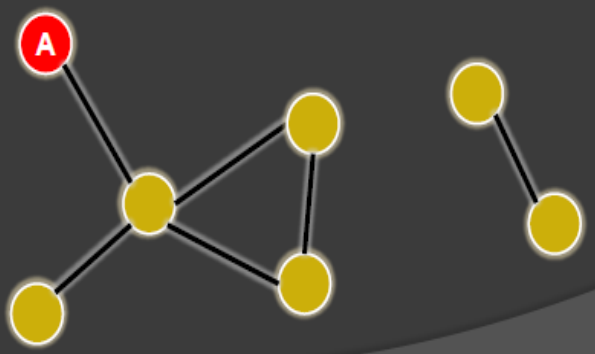
1. Każdy węzeł grafu oznacz *jako nieodwiedzony*.
2. Dla każdego *nieodwiedzonego* wierzchołka należy *odwiedzić* ten węzeł startowy i węzły z nim połączone:

a) Oznacz węzeł startowy jako *odwiedzony* oraz wprowadź go do *pustej kolejki typu FIFO*

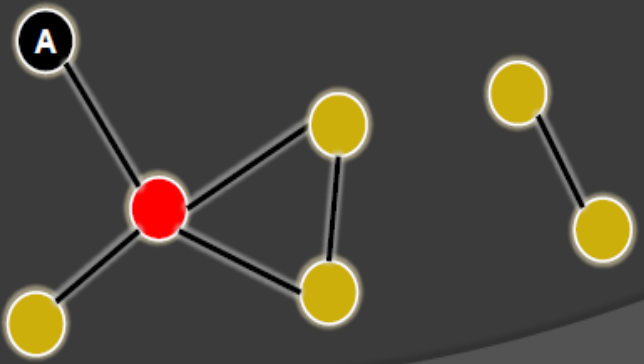
b) Dopóki kolejka FIFO nie jest pusta:

- Wyjmij wierzchołek *z przodu kolejki*
- Wszystkie *nieodwiedzone* następniki tego wierzchołka oznacz jako *odwiedzone*, a następnie wstaw sekwencyjnie do kolejki. Węzeł wyjęty z kolejki oznacz jako *przetworzony*.
- Jeśli poszukiwany element odpowiada temu węzłowi, to prezentuj wynik, STOP.

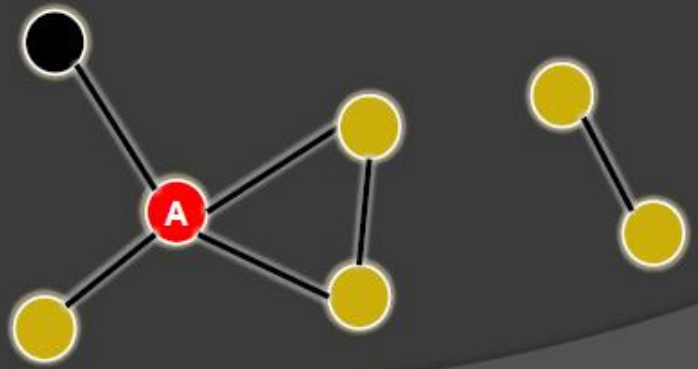
- Wybieramy dowolny nieodwiedzony wierzchołek jako wierzchołek ***startowy***
- Staje się on wierzchołkiem ***aktualnym*** (A)
- Oznaczamy go jako ***odwiedzony*** ( )



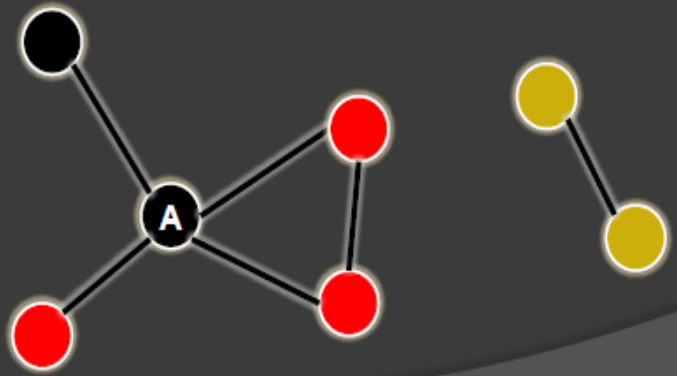
- Oznaczamy jako odwiedzone, po kolei, wszystkie następniki wierzchołka aktualnego
- Wierzchołek aktualny staje się przetworzonym



- Przechodzimy do następnego, w kolejności odwiedzania, wierzchołka

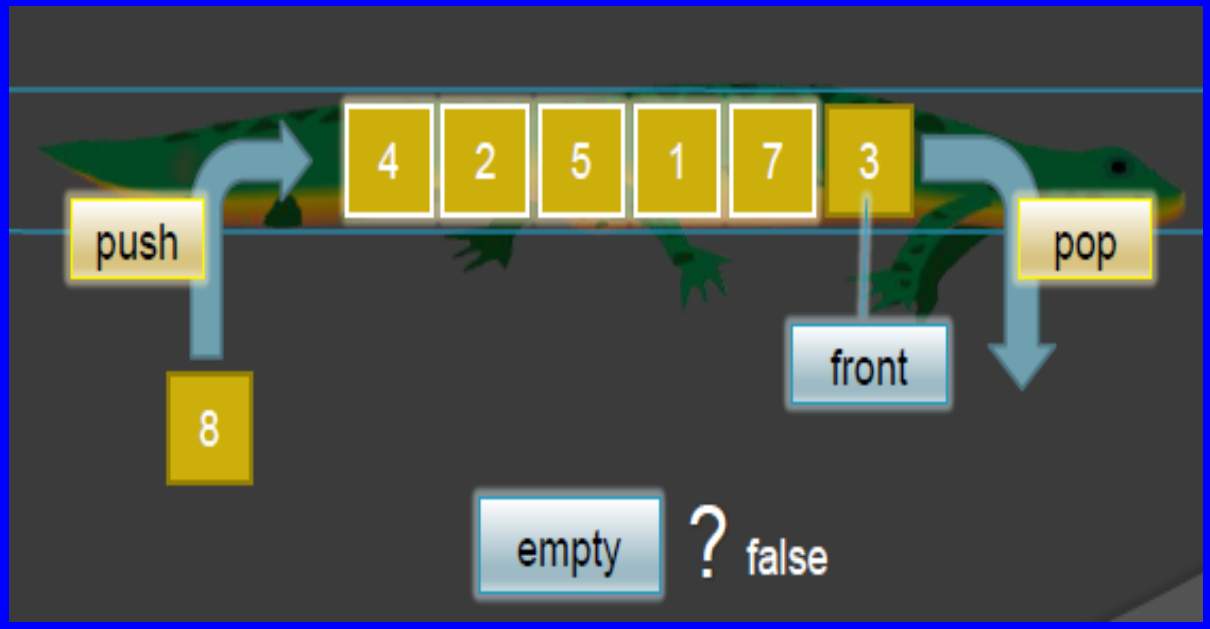


- Oznaczamy jako odwiedzone, po kolei, wszystkie następniki wierzchołka aktualnego
- Wierzchołek aktualny staje się przetworzonym



Kolejka

- *Podstawowa struktura danych w implementacji algorytmu BFS*
- *Zapewnia przechodzenie do wierzchołków w kolejności ich wcześniejszego oznaczania jako odwiedzone*

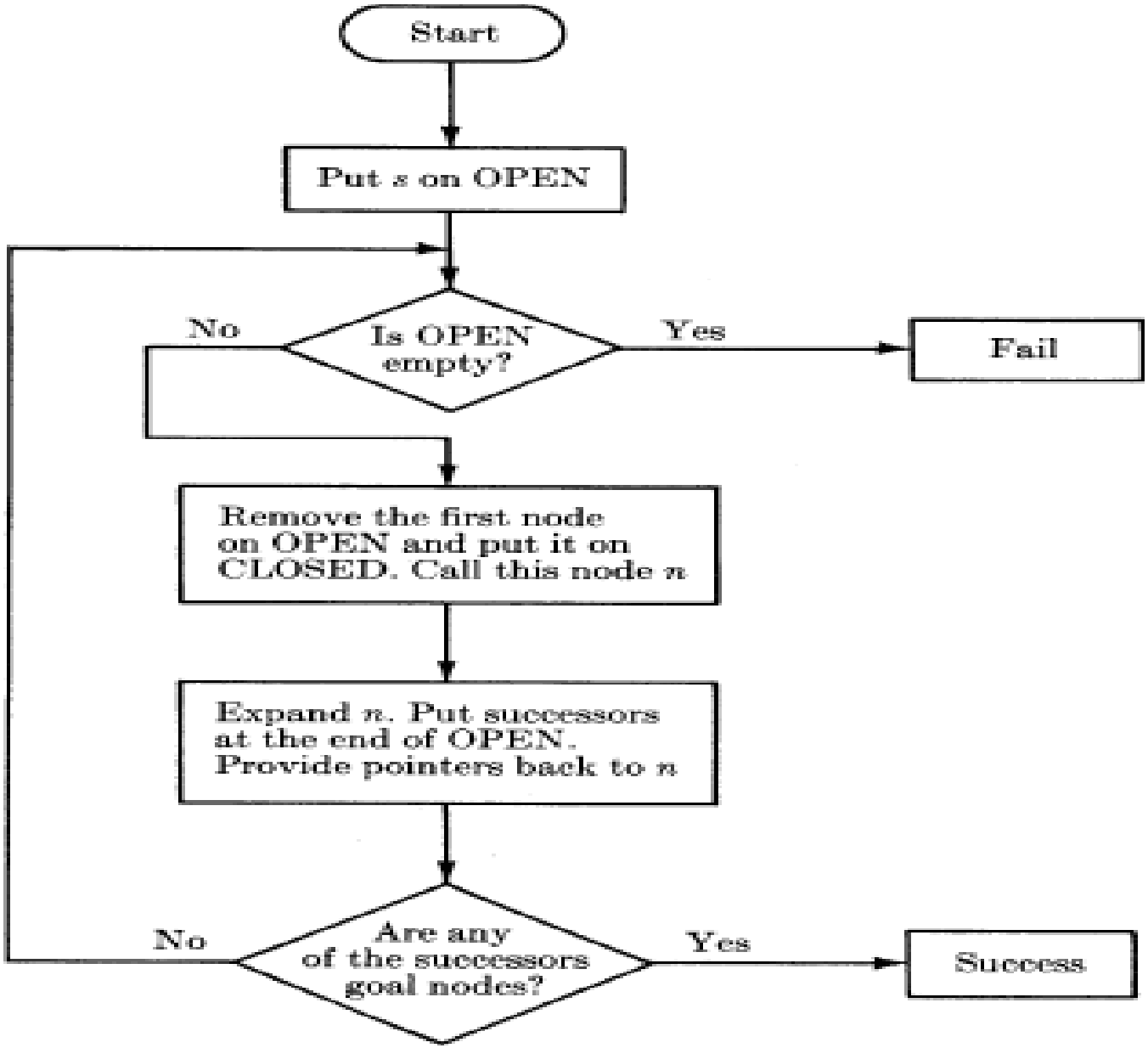


```
#include <queue>          // dołączenie klasy „kolejka”
using namespace std;

const int maxn = 100;    // m – liczba krawędzi
int n, m;                // n – liczba wierzchołków
bool  graf[maxn][maxn]; // graf – tablica przyległości węzłów
bool  stan[maxn];        // stan – tablica stanów wizytowania
                         //                wierzchołków
void szukanie(int start)  {
queue<int> q;
stan[start] = true;      // ocechowanie węzła startowego
q.push(start);           // węzeł start do pustej kolejki
      while(!q.empty()) {  //
      int u;
      u = q.front();     // pobranie węzła z kolejki FIFO
      q.pop();
      for (int i= 0; i< n; ++i)          // odwiedzenie następnika
           if (graf[u][i] && !stan[i]) {  //    nieodwiedzonego
           stan[i] = true;               // ocechowanie
           q.push(i);    }               // na koniec kolejki
       }
 }
```

```
algorytmDFS() {
        for (int i= 0; i< n; ++i)  // początkowo węzły
              stan[i] = false;       //  są nieodwiedzone
        for (int i= 0; i< n; ++i)
              if (!stan[i]) {            // rozpoczęcie szukania
                    szukanie(i);         // „wszerz” dla węzła i
               }
}
```

Szukanie wszerz



Szukanie wszerz - złożoność obliczeniowa

Szacowana liczba węzłów dla drzewa:

$$|V| = 1 + b + b^2 + b^3 + \ldots + b^d$$

gdzie:

b - liczba węzłów potomnych generowanych z jednego węzła *(współczynnik rozgałęzienia)*

d - liczba przebytych warstw (wysokość drzewa, długość ścieżki)

Złożoność obliczeniowa metody $O(b^d) \sim O(|V|+|E|)$

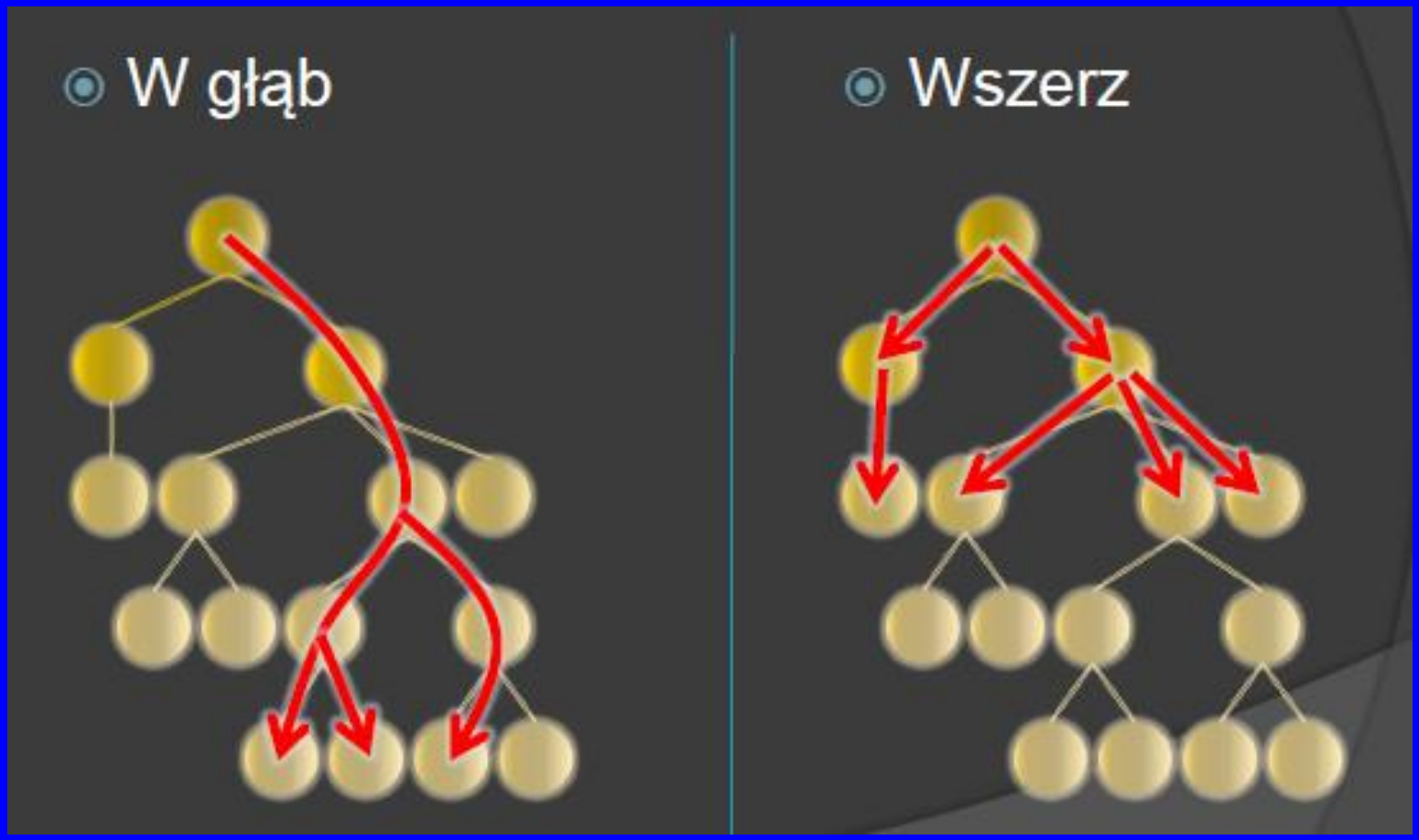
Szukanie wszerz

Głębokość	Węzły	Czas		Pamięć	
0	1	1	ms	100	bajtów
2	111	0.1	s	11	kB
4	11111	11	s	1	MB
6	10^6	18	m	111	MB
8	10^8	31	godz	11	GB
10	10^{10}	128	dni	1	TB
12	10^{12}	35	lat	111	TB
14	10^{14}	3500	lat	11111	TB

Wymagany czas i pamięć przy szukaniu metodą wszerz: $b = 10$, 1000 węzłów/sekundę, 100 bajtów na zapis węzła

Szukanie w głąb

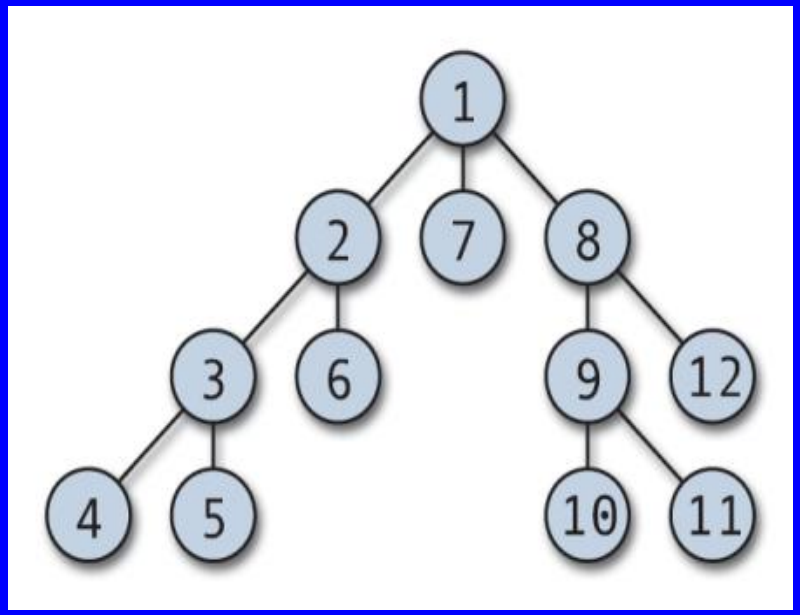
Przeszukiwanie zaczyna się od korzenia i porusza się w dół do samego końca gałęzi, po czym wraca się o jeden poziom i próbuje kolejne gałęzie

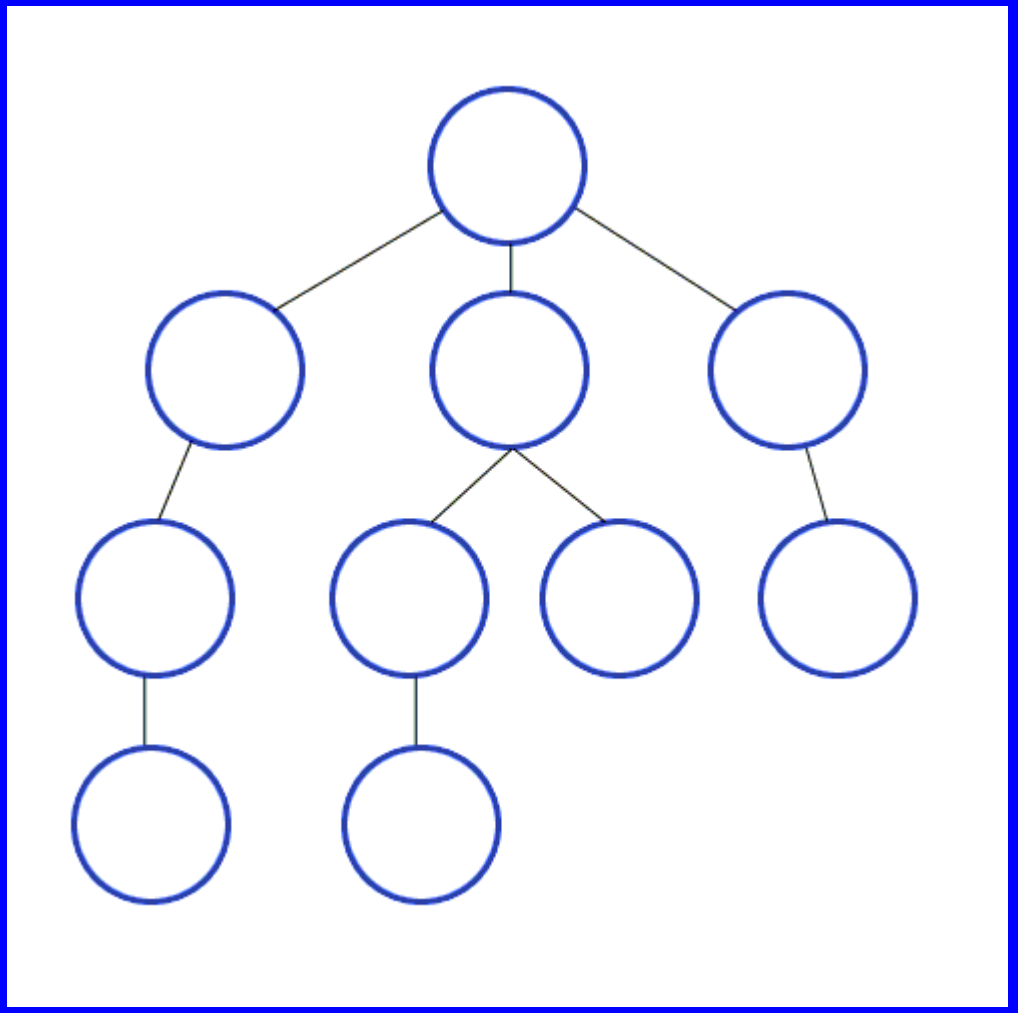


Szukanie w głąb *(ang. Depth-First Search, DFS):*

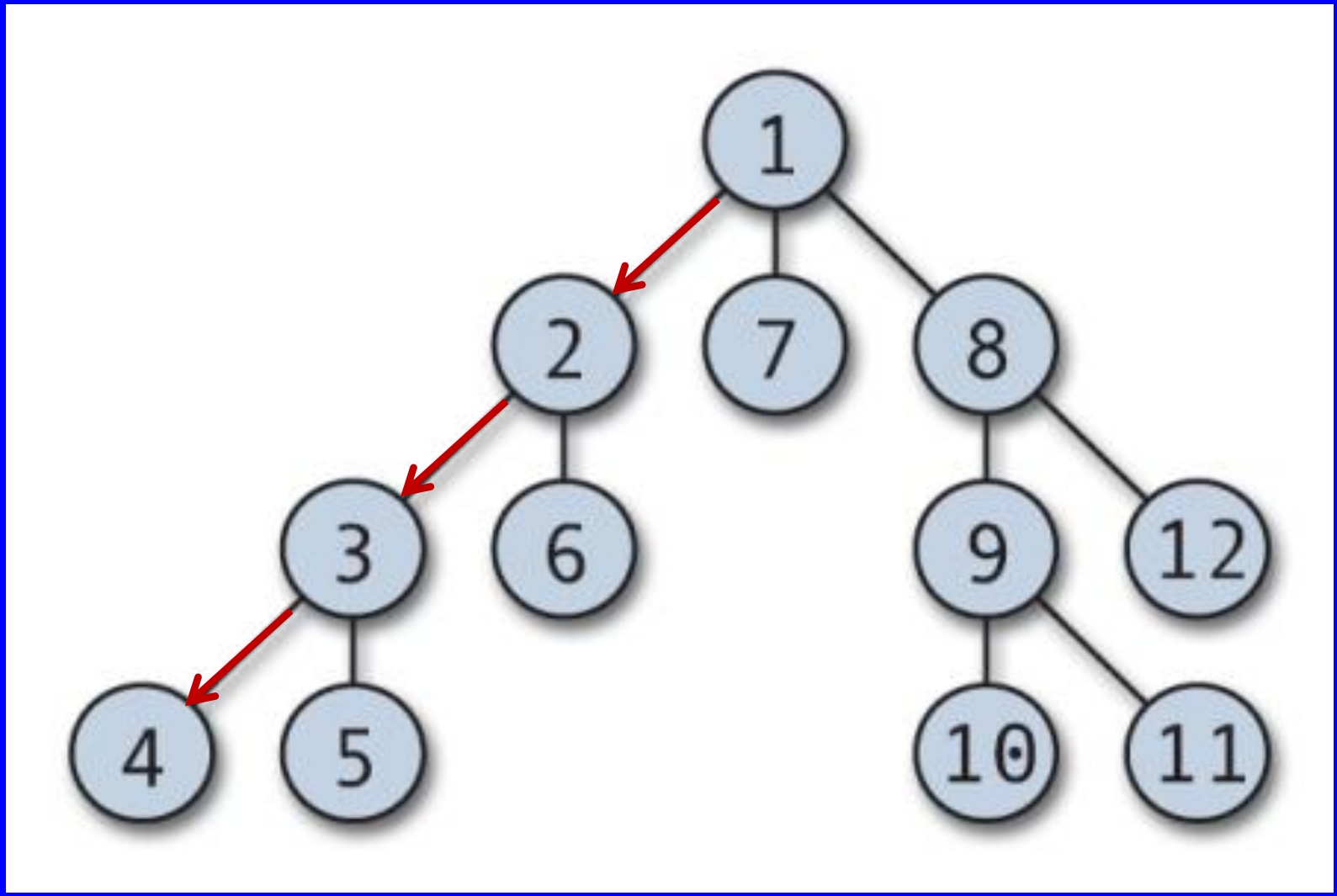
- **Wyznaczania najkrótszych ścieżek między dwoma wierzchołkami w drzewie**
- **Sprawdzenia, czy istnieje ścieżka między dwoma wierzchołkami w grafie**
- **Wyznaczania spójnych składowych, a także silnie spójnych składowych w grafie skierowanym**
- **Sortowanie topologiczne skierowanego grafu acyklicznego**
- **W algorytmach typu *brute force*, np. do przeglądania drzewa gry (*min-max* czy też *alpha-beta*)**

Szukanie w głąb
(ang. Depth-First Search, DFS):

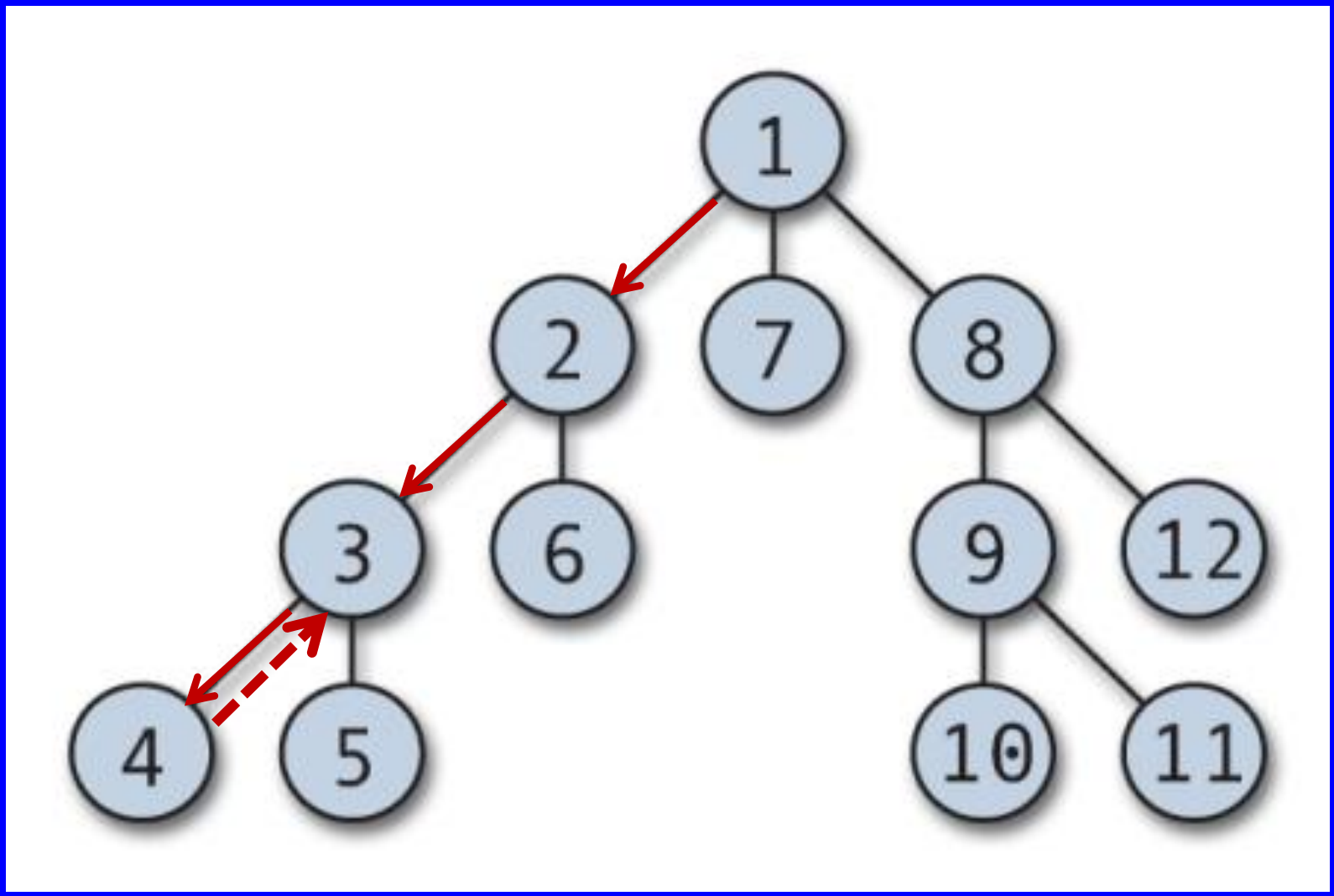




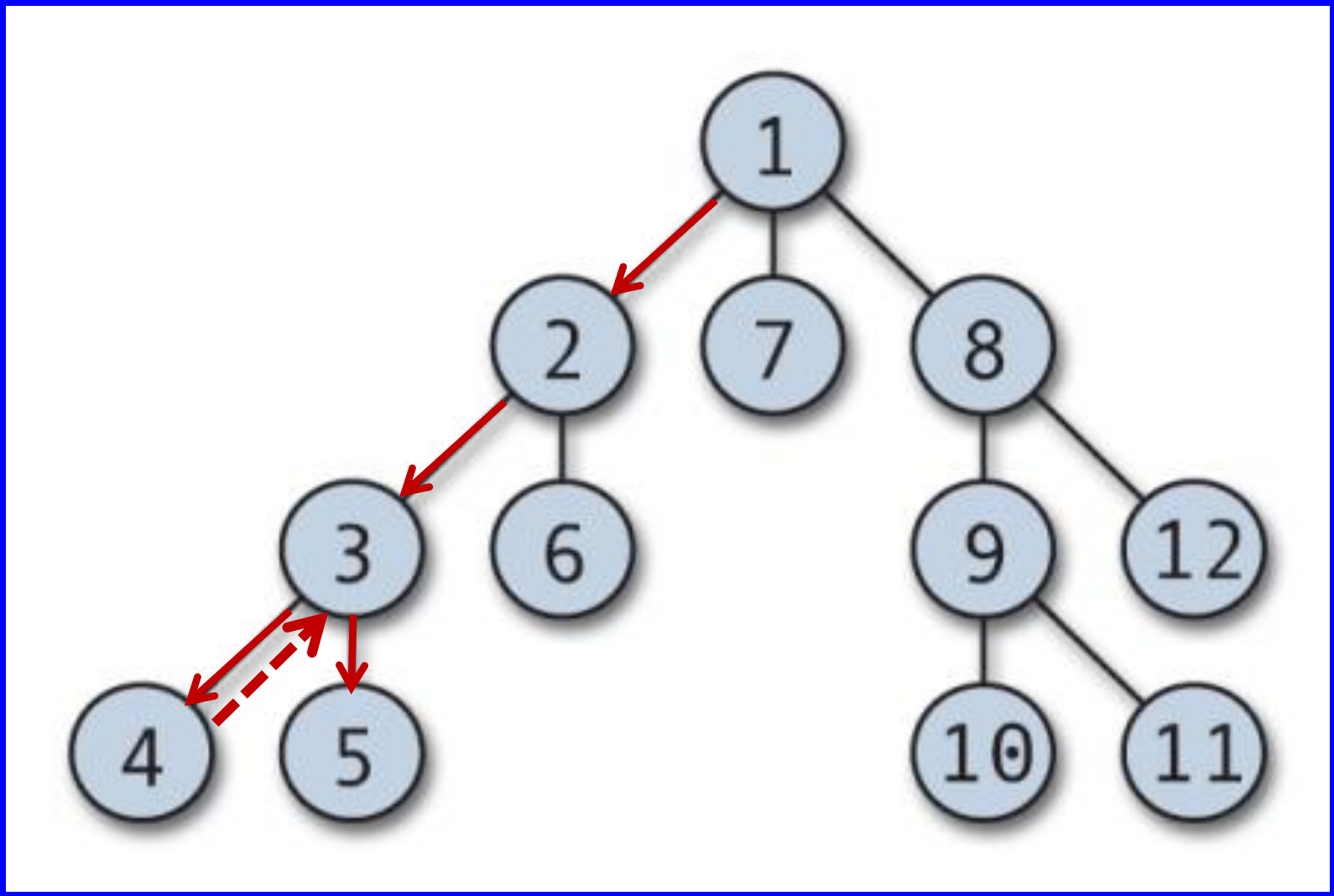
Odwiedzanie i powroty



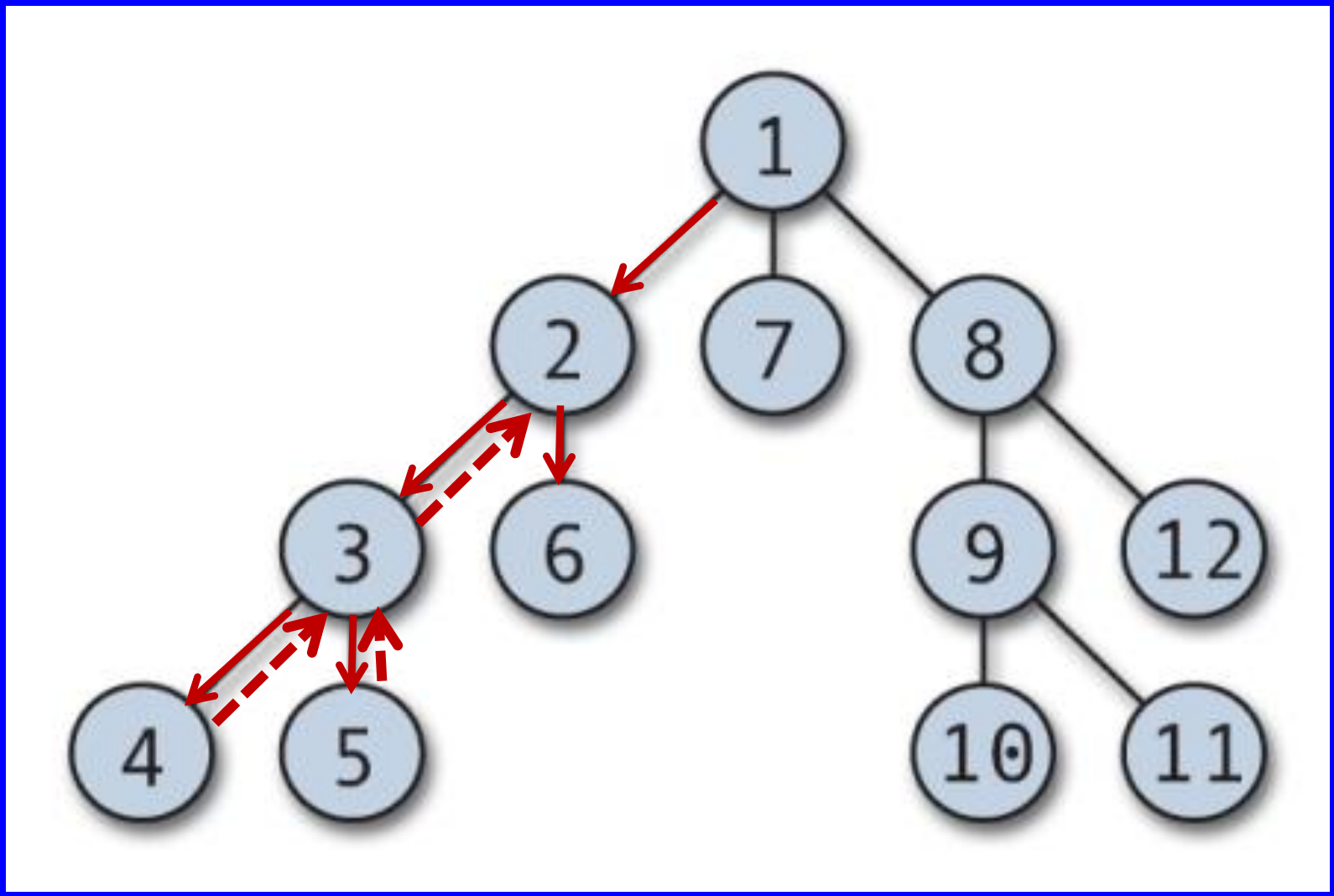
Odwiedzanie i powroty



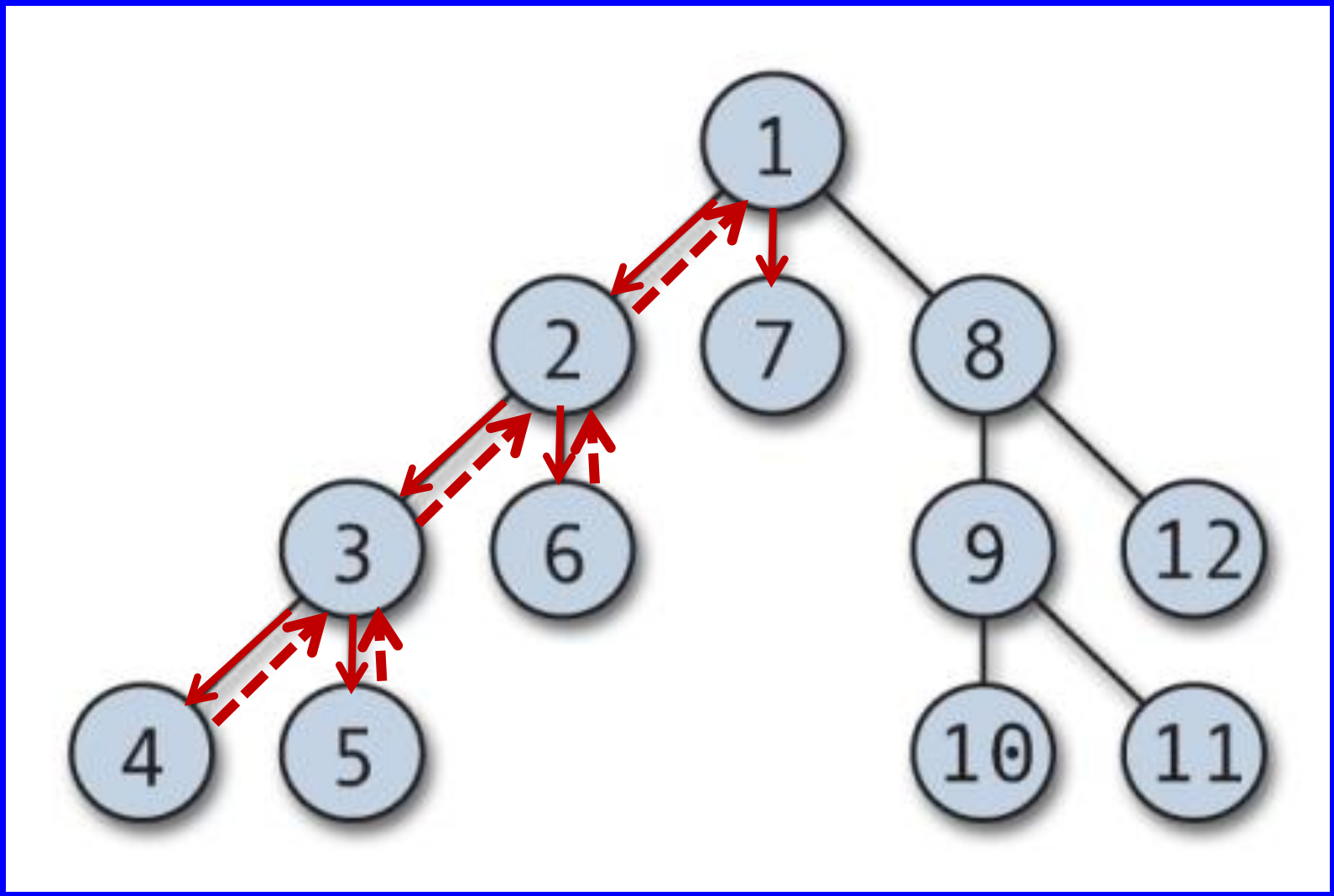
Odwiedzanie i powroty



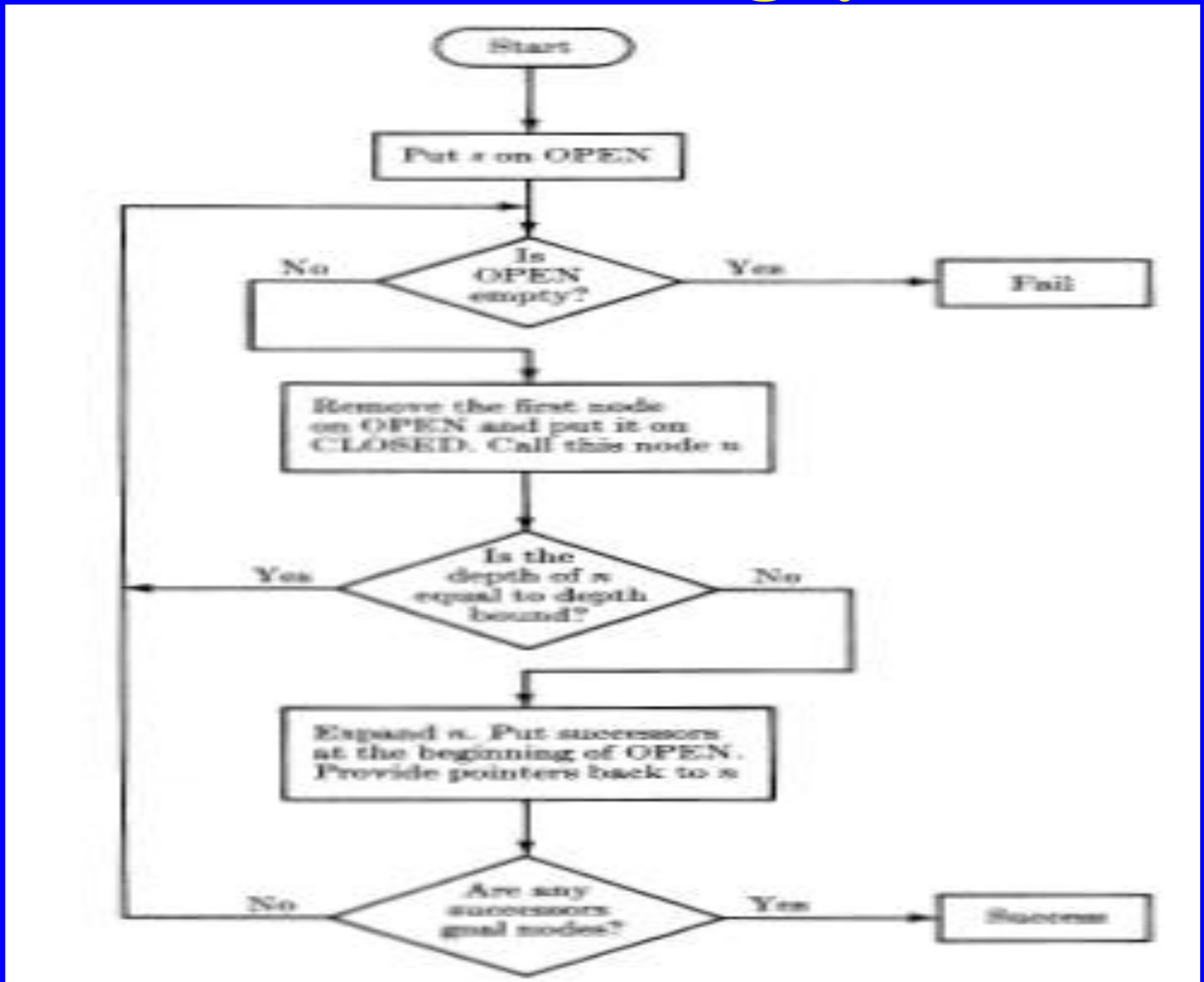
Odwiedzanie i powroty



Odwiedzanie i powroty



Szukanie w głąb



Algorytm DFS

1. Każdy wierzchołek oznacz jako *nieodwiedzony*
2. Dla każdego wierzchołka *nieodwiedzonego* należy wybrać reprezentanta *x*, oznaczyć go jako *odwiedzony* oraz skonstruować *ścieżkę* do wybranego *liścia* drzewa.
   - *REKURENCJA(x) // konstruowanie ścieżki*
     Jeśli dla wierzchołka *x* istnieje krawędź do wierzchołka *y*, który nie był jeszcze *odwiedzony*, to oznacz wierzchołek *y* jako *odwiedzony* i wywołaj rekurencyjnie funkcję *REKURENCJA*, przyjmując *y* jako wierzchołek startowy *x*

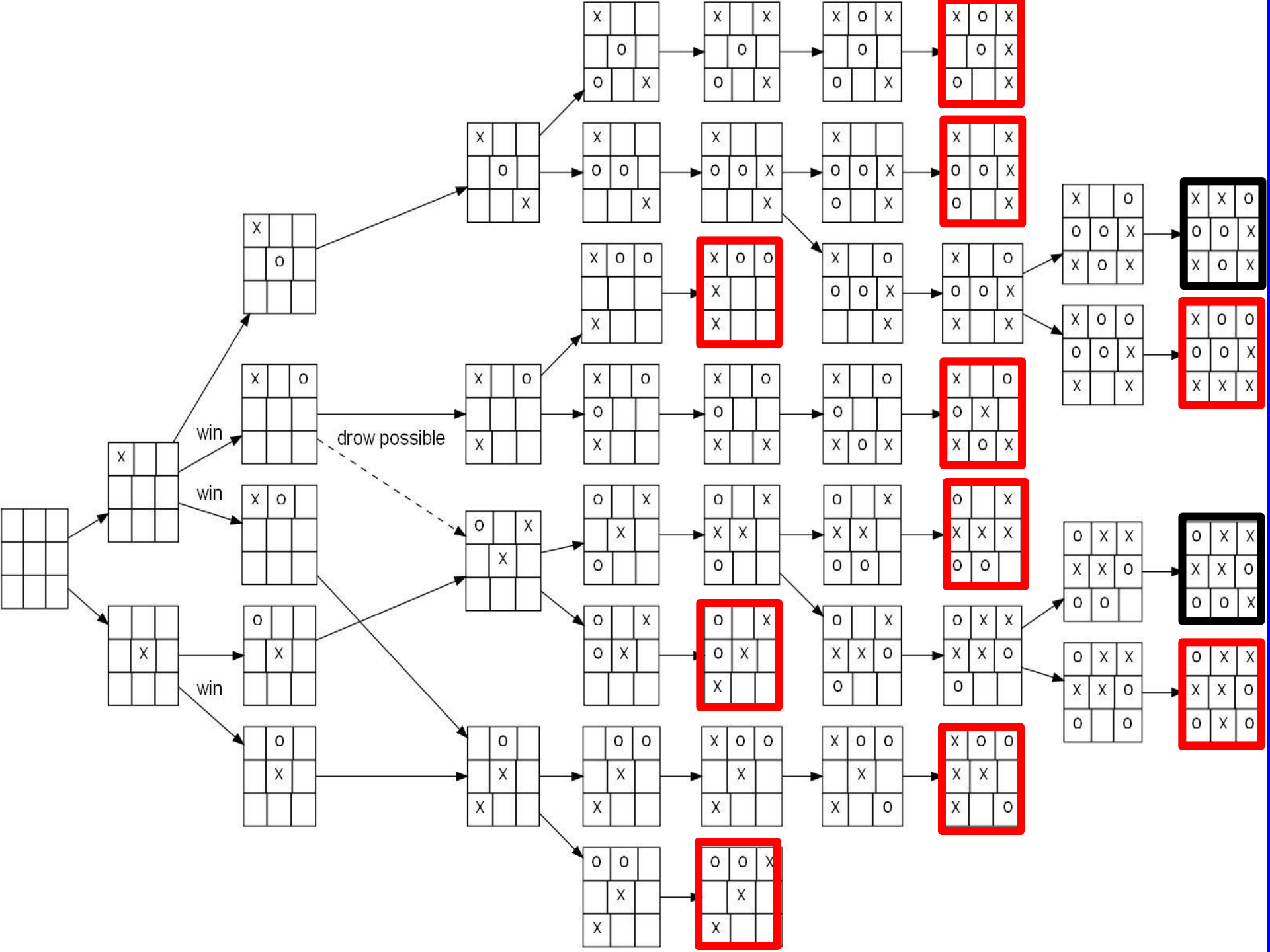
Złożoność pamięciowa i obliczeniowa

Złożoność pamięciowa przeszukiwania w głąb w przypadku drzewa jest o wiele mniejsza niż przeszukiwania wszerz, gdyż algorytm wymaga zapamiętania tylko ścieżki od korzenia do bieżącego węzła (*d* – długość ścieżki): *O(d)*

Przeszukiwanie wszerz wymaga zapamiętywania wszystkich węzłów w danej odległości od korzenia, co zwykle rośnie wykładniczo w funkcji długości ścieżki $O(b^d)$

Złożoność obliczeniowa obu algorytmów

$$O(|V|+|E|) \sim O(b^d)$$


win
win
win
drow possible