

Análise de grafos parte 3: métodos de percurso

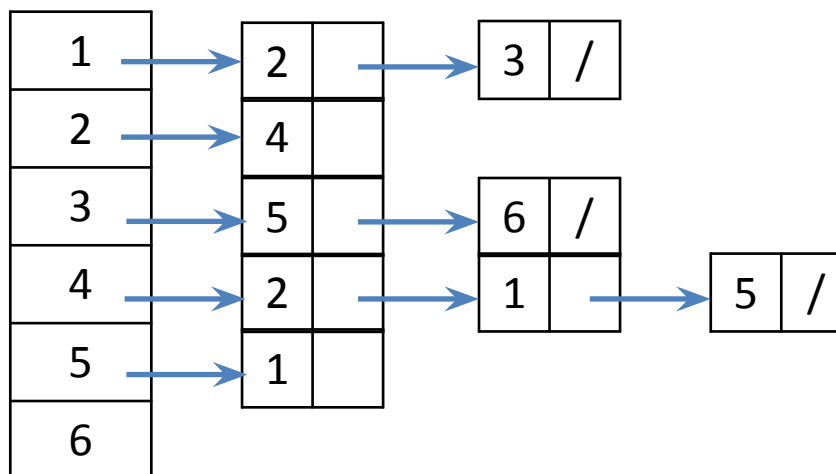


Retomando...

- Matriz de incidência;

	1	2	3	4
1	1	0	0	0
2	0	0	-1	1
3	-1	1	1	0
4	0	0	0	0

- Lista de adjacência.



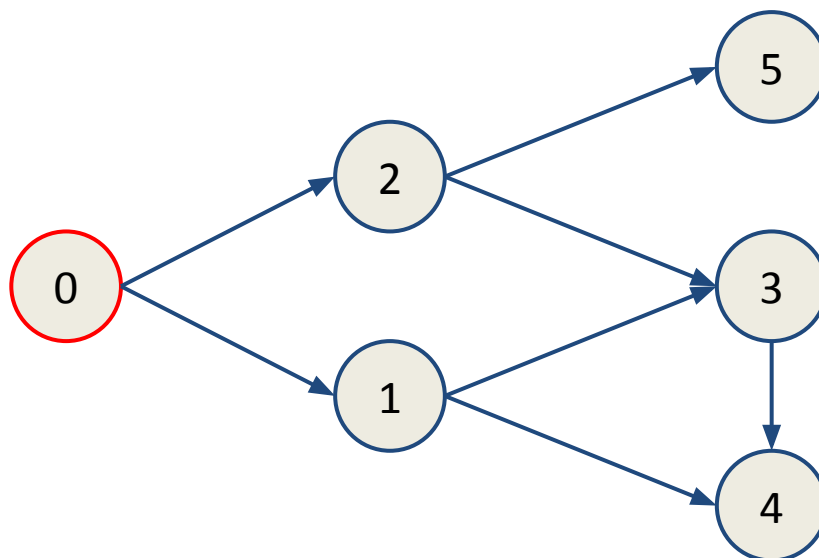


Percursos em grafos

- Visitar todos os nós de um grafo de acordo com uma ordem pré-estabelecida, onde:
 - Visitar é processar ou manipular as informações dos nós (imprimir, calcular, etc.).
- Duas formas principais de percurso em grafos:
 - Profundidade;
 - Amplitude.
- Semelhante aos percursos sobre árvores binárias;
- Em ambos os métodos, deve-se escolher arbitrariamente um nó para iniciar o percurso e então analisar seus nós adjacentes.

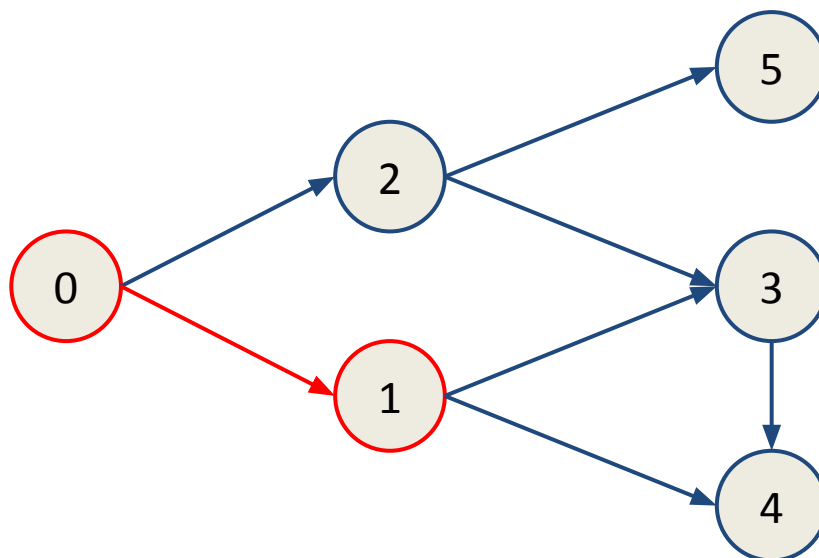
Percurso em profundidade

- Escolher um nó inicial: 0



Percurso em profundidade

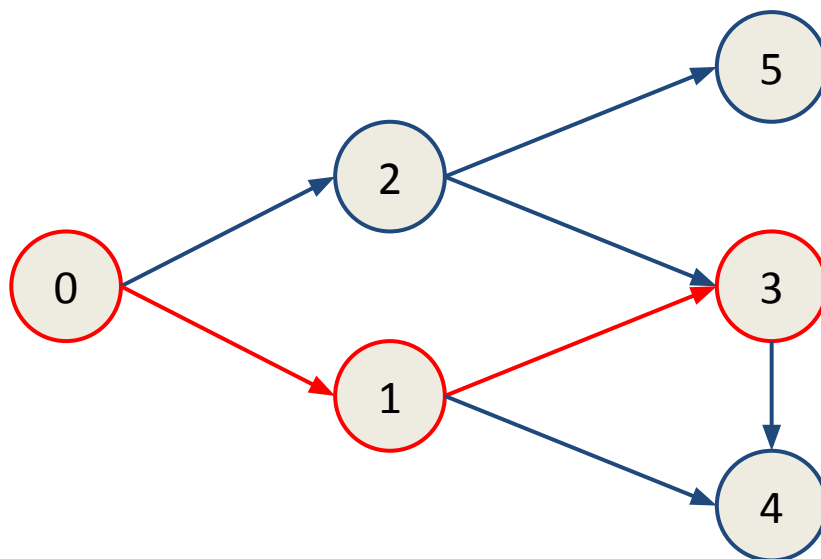
- Visitar o sucessor de 0: 1





Percurso em profundidade

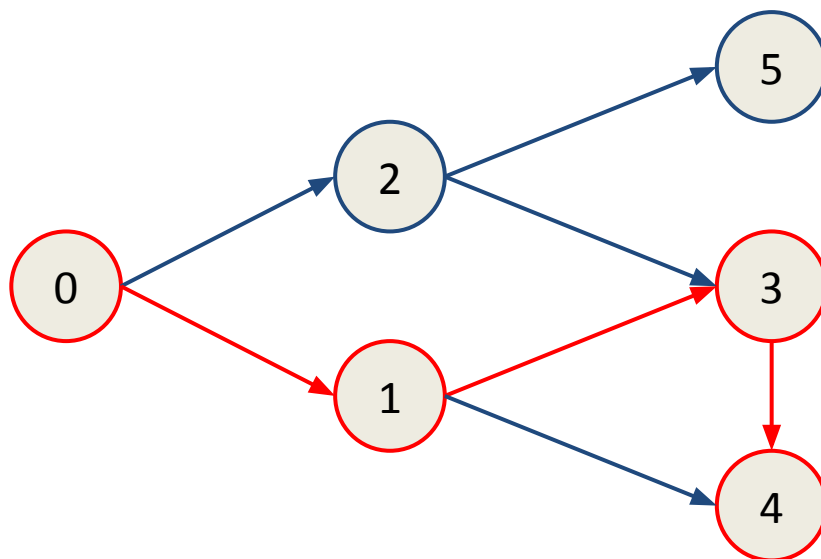
- Visitar o sucessor de 1: 3





Percurso em profundidade

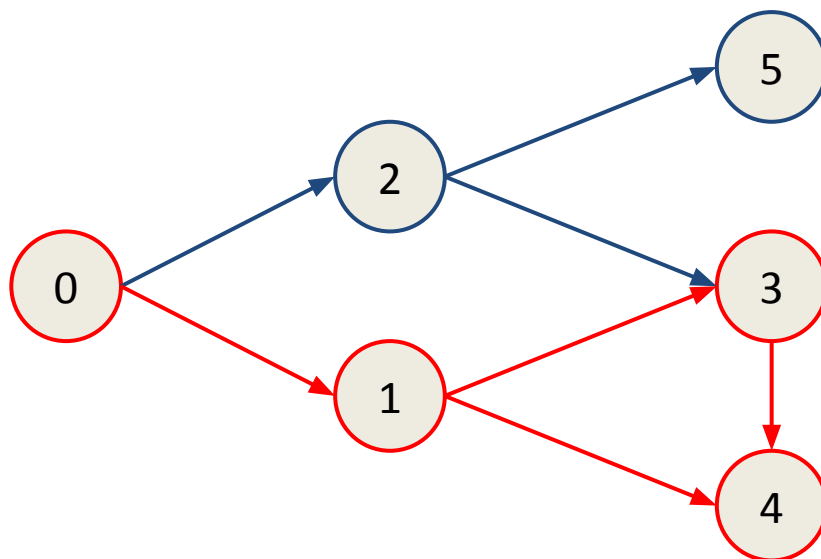
- Visitar o sucessor de 3: 4





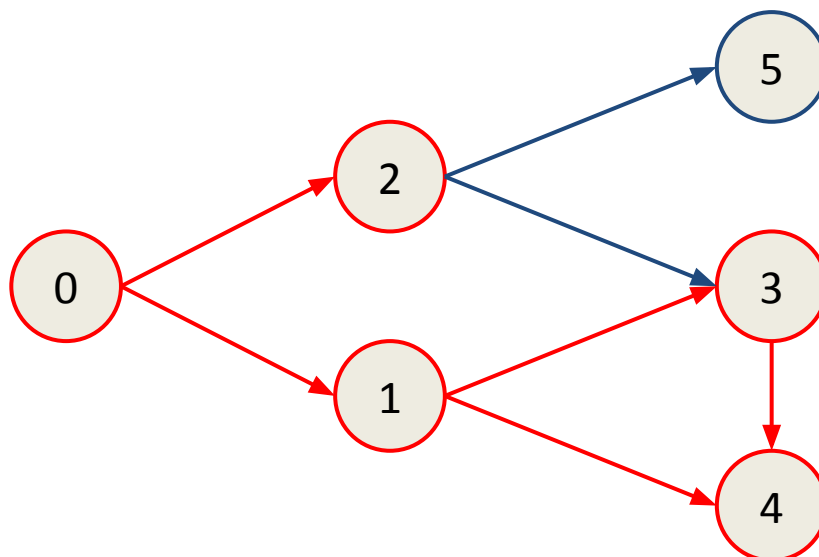
Percurso em profundidade

- O outro sucessor de 1 já foi visitado: 4



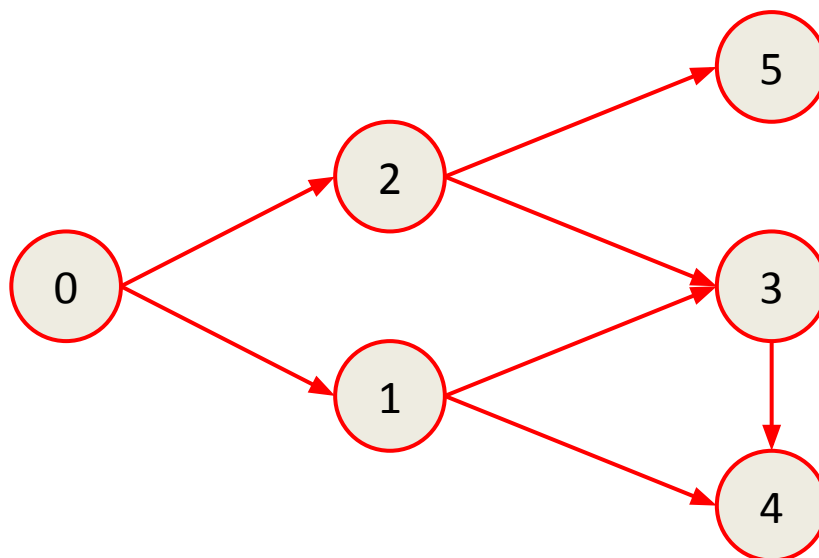
Percurso em profundidade

- Volta para o primeiro nó anterior ao atual que possua nós não visitados:
 - Neste caso, volta ao nó inicial (0) para visitar seu outro sucessor: 2



Percurso em profundidade

- O primeiro sucessor de 2 já foi visitado: 3;
- Visita-se o próximo sucessor de 2: 5.



Percurso em profundidade

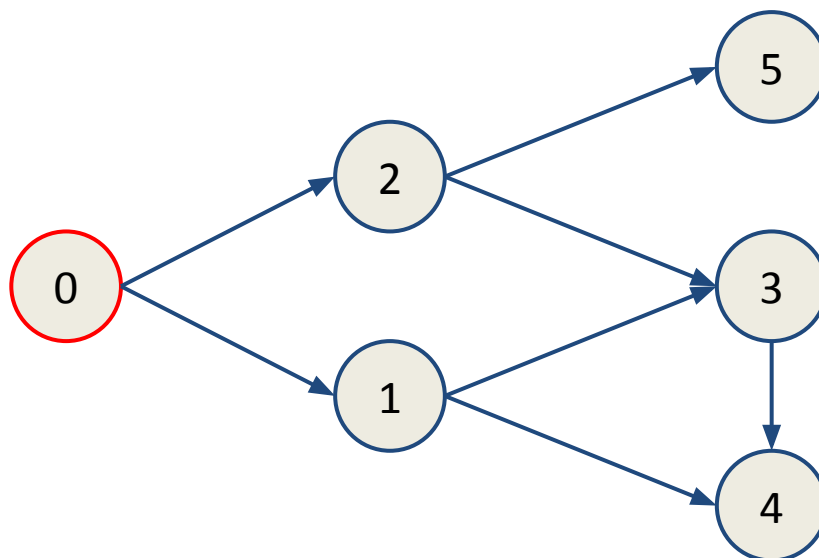
- Visita-se o nó *atual*;
- Coloca-se o nó em uma pilha;
- *atual* = sucessor (i)

```
1 void profundidade(int totalNos, int atual, int visitados[])
2 {
3     int i;
4     printf("%d ", atual); // manipula nó atual
5     visitados[atual] = 1;
6     for (i=0; i<totalNos; i++)
7     {
8         if(mat[atual][i] == 1 && !visitados[i])
9         {
10             profundidade(totalNos, i, visitados);
11         }
12     }
13 }
```

Para implementação com lista de adjacência ou matriz de incidência, alterar essa verificação.

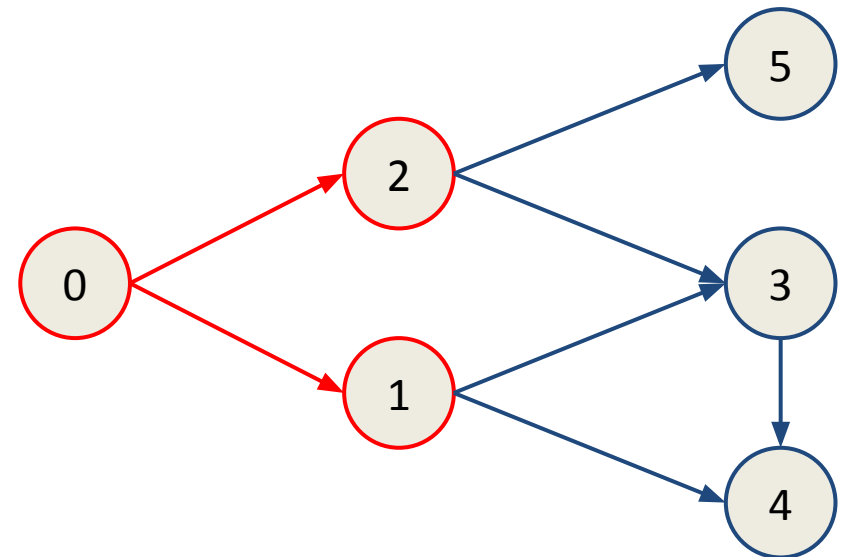
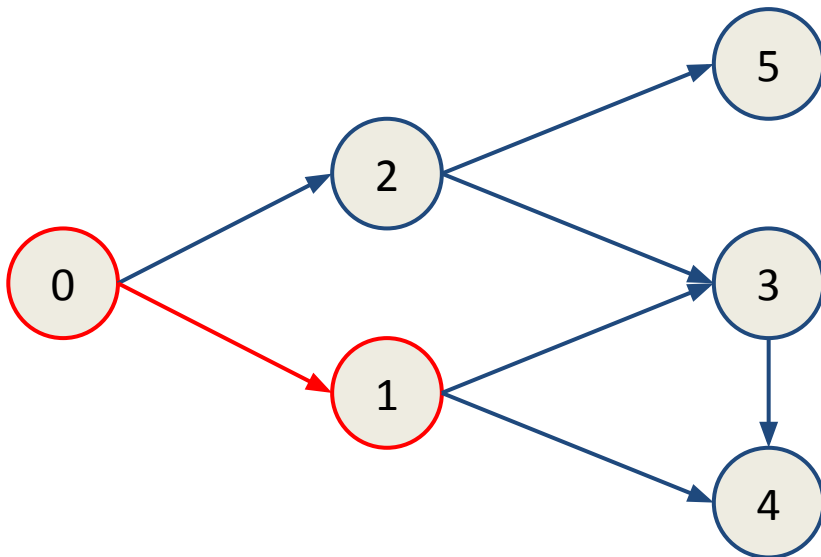
Percurso em amplitude

- Escolher um nó inicial:
 - Atual = 0;



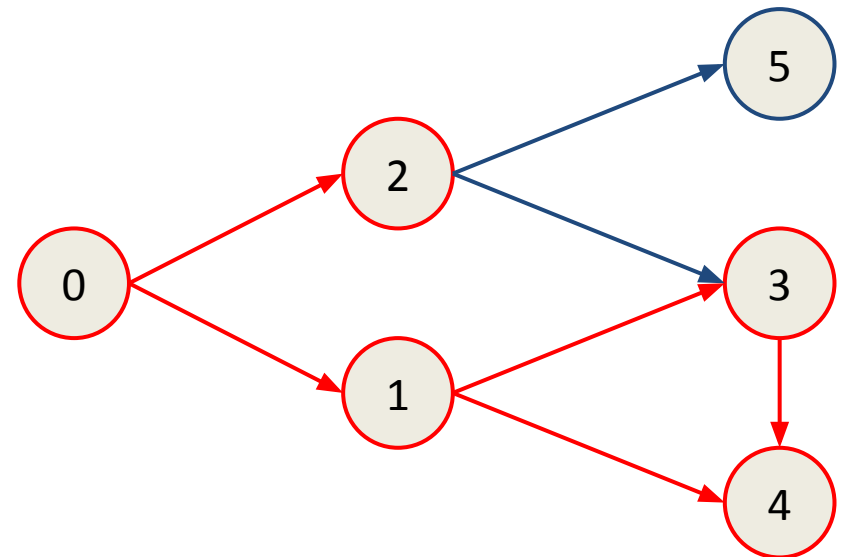
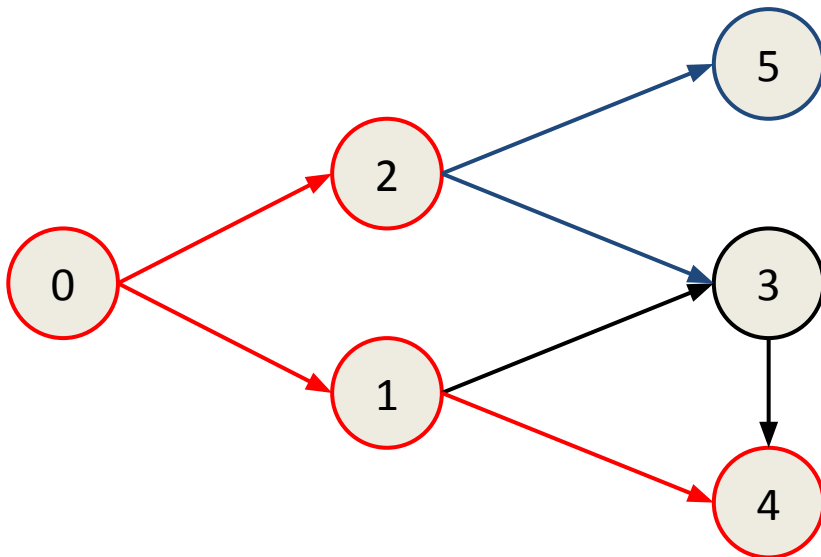
Percurso em amplitude

- Atual = 0;
- Visitar os adjacentes (sucessores) do atual: 1 e 2.



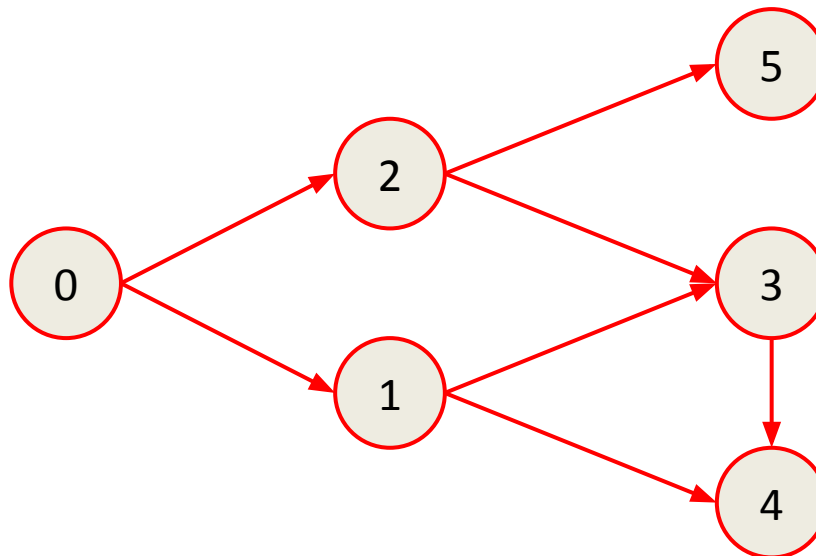
Percurso em amplitude

- Atual = 1;
- Visitar os adjacentes (sucessores) do atual: 3 e 4.



Percurso em amplitude

- Atual = 2;
- Visitar sucessores do atual: 5. O nó 3 já foi visitado;
- Atual = 3: Nó 4 já havia sido visitado;
- Nós 4 e 5 não possuem sucessores.



Percurso em amplitude ou largura

- Visita-se o nó *atual*;
- Coloca-se o nó em uma fila;
- Ao final, *atual* = próximo elemento da fila e repete-se o processo.

```
1 void amplitude (int atual, int visitados[])
2 {
3     int i;
4     printf("%d ", atual); // manipula nó atual
5     visitados[atual] = 1;
6     while(atual != -1)
7     {
8         for (i=0; i<6; i++)
9         {
10             if(grafo[atual][i] == 1 && !visitados[i])
11             {
12                 printf("%d ", i); // manipula nó
13                 visitado[i] = 1;
14                 incluir(i); // na fila
15             }
16         }
17         atual = retirar(); // da fila. Se -1, então fila vazia
18     }
19 }
```




Exercício sobre percursos

- Para que os algoritmos apresentados na aula funcionem corretamente é preciso adicionar outras funções. Dessa forma:
 - Implemente um algoritmo para percorrer um grafo utilizando o método por profundidade E por amplitude;
 - Fica a critério do aluno criar os dois métodos de percurso no mesmo arquivo ou separadamente;
 - Pesquisar sobre algoritmos para encontrar o menor caminho em grafos.