DCC206 – Algoritmos 1

Aula 01 – Introdução a Grafos

Professor Renato Vimieiro

DCC/ICEx/UFMG

Introdução

- Muitas aplicações envolvem a modelagem de conexões entre pares de elementos
- Exemplos:
  - Navegação: Conhecer a rota mais curta entre dois pontos da cidade?
  - Web: Páginas contêm links para outras páginas.
  - Circuitos elétricos: Desenho do circuito integrado contém curto-circuito? É possível desenhar um determinado circuito em um chip sem que haja curto-circuito?
  - Agendamento de tarefas: Um processo para ser executado depende de várias tarefas não independentes entre si. Em que ordem as tarefas devem ser executadas?

Introdução

- Exemplos (continuação):
  - Redes de computadores: Entender a estrutura da rede para dimensionar a infraestrutura, encontrar pontos críticos da rede (se caírem desconectam a rede)
  - Atribuição de pessoas a cargos: Diversas pessoas competem por diferentes cargos, como designá-las aos cargos da melhor maneira possível?
  - Redes sociais: Pessoas seguem amigos. Descobrir grupos de pessoas com interesses em comum. Entender processos sociais…

Definições

- Um grafo é formalmente definido por um par
  - G = (V,E)
  - V é um conjunto de **vértices** (elementos sendo representados)
  - E é um conjunto de **arestas** (conexões entre vértices)
- Os elementos do conjunto de vértices são, em geral, representados por inteiros entre 0 e |V|-1
- Uma aresta *e* é um conjunto de dois vértices *{u,v}* que são chamados de **terminais** (extremidades) da aresta. Da mesma forma, *e* é chamada de **incidente** nos vértices *u* e *v.*
- Se existe aresta entre u e v, eles são chamados de **adjacentes**
- As arestas podem ou não ter pesos associados para quantificar a relação

Grafos

- Exemplo (GoT):
- V = personagens
- E = relacionam-se

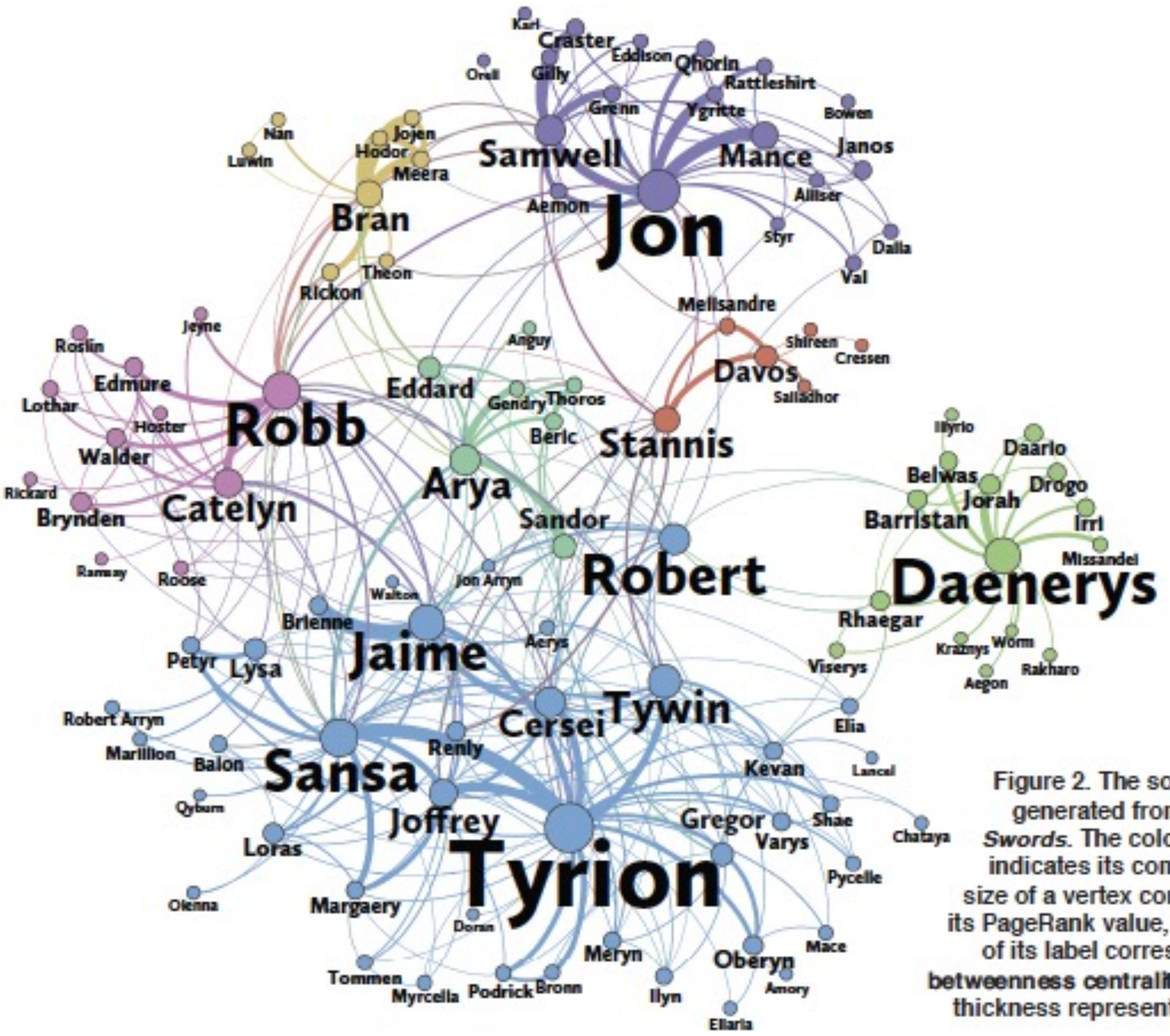


Figure 2. The social network generated from *A Storm of Swords*. The color of a vertex indicates its community. The size of a vertex corresponds to its PageRank value, and the size of its label corresponds to its betweenness centrality. An edge's thickness represents its weight.

Figura retirada de Beveridge e Shan (2016)

Grafos não-direcionados

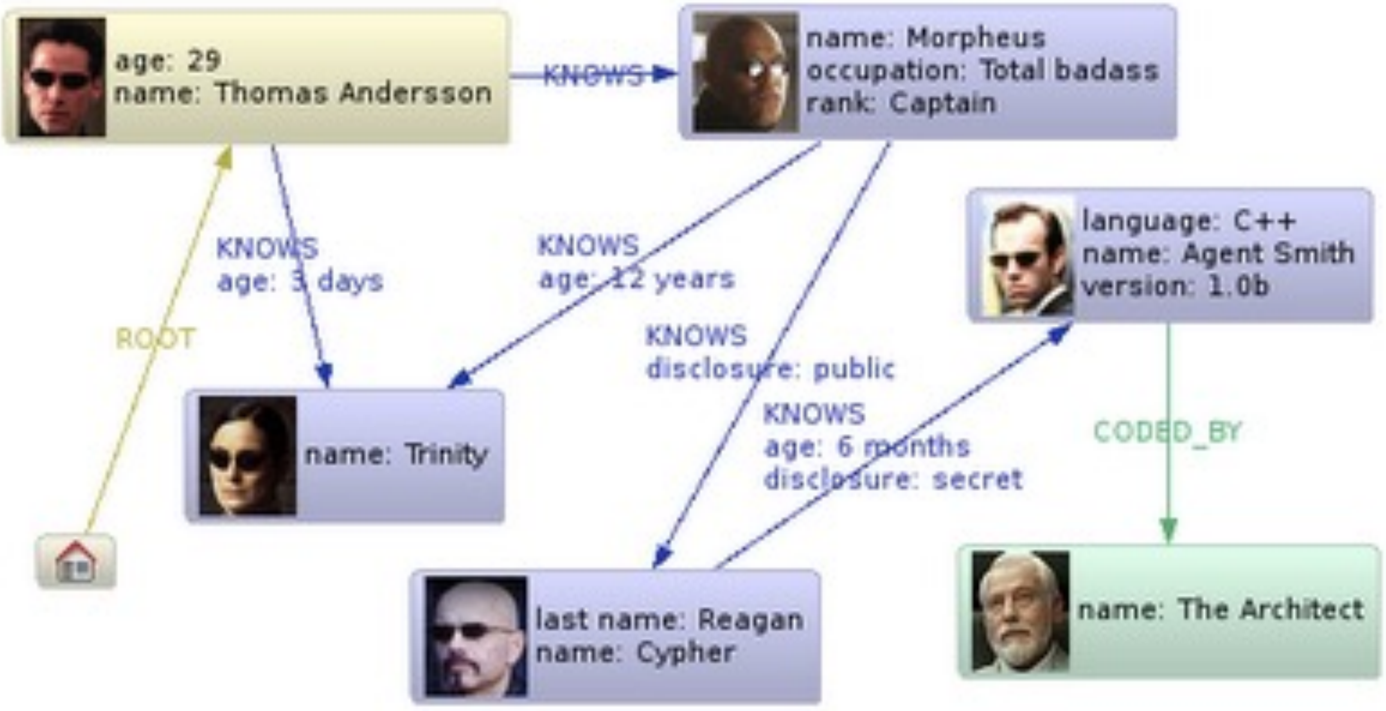
- A definição de grafos como apresentada permite que exista somente uma aresta entre dois vértices
  - Pode-se relaxar tal definição para acomodar múltiplas arestas entre pares de vértices. Essas arestas são chamadas de **arestas paralelas**.
- A definição também permite a relação de um vértice com ele mesmo
  - Essas arestas são chamadas de **(self-)loops**.
- Um grafo é chamado de **simples** se ele não possui loops nem arestas paralelas
  - Assumiremos que um grafo é simples a menos que seja dito o contrário
  - Grafos com arestas paralelas ou self-loops são chamados de **multigrafos**

Grafo direcionado

- Em alguns casos, as relações entre os elementos não são simétricas
  - Por exemplo: x é divisor de y; ao se vestir, a meia deve ser vestida antes da calça (peças de roupas são vértices e arestas determinam a precedência)
- Para esses casos, a definição de grafos é modificada
  - Arestas são pares ordenados de vértices
  - Logo $(u, v) \in E \nrightarrow (v, u) \in E$
- Esses grafos são conhecidos como **grafos direcionados** ou dígrafos
  - As arestas tem direção *u* é chamado de origem e *v* de destino

Grafo direcionado

- Exemplo (Matrix)
- V = personagem
- E = A conhece B



Fonte: http://imasters.com.br/artigo/23554/banco-de-dados/diversao-com-neo4j-e-jruby?trace=1519021197&source=single

Grau de vértices

- O número de arestas incidentes em um vértice é chamado de **grau** do vértice
  - $\sum d(v) = 2|E|$
  - Número de vértices de grau ímpar é sempre par!
- Grafos direcionados:
  - grau de entrada (in-degree): número de arestas em que v é destino;
  - grau de saída (out-degree): número de arestas que tem v como origem

Caminhos e ciclos

- Um **caminho** em um grafo é uma sequência de vértices adjacentes entre si
  - P = v1-v2-v3-…-vk
  - $(v_i, v_{i+1}) \in E$
  - O comprimento de um caminho é a quantidade de arestas percorridas entre o primeiro e último vértices
- Se existe caminho entre *u* e *v, v* é dito alcançável a partir de *u*
- Se u=v, então o caminho é chamado de **ciclo**/circuito
- Um caminho/ciclo é **simples** se todos os seus vértices são distintos
- A **distância** entre dois vértices u e v é o tamanho do menor caminho entre eles

Conectividade em grafos

- Um grafo não-direcionado é **conectado** se, para quaisquer vértices u e v, u é alcançável a partir de v.
  - Se o grafo for desconectado (desconexo), cada grupo de vértices alcançáveis entre si formam um **componente conexo**
- Um grafo direcionado é **fortemente conexo** se quaisquer pares de vértices forem mutuamente alcançáveis
  - Os **componentes fortemente conexos** são formados por grupos de vértices mutuamente alcançáveis entre si

Grafos completos e bipartidos

- Um **grafo completo** é um grafo cujos vértices são todos adjacentes entre si
  - Denota-se um grafo completo com *n* vértices por K_n
  - O número de arestas é |V|(|V|-1)/2
- Em algumas situações, o conjunto de vértices está particionado pela natureza do problema
  - Atribuição de pessoa a cargos: dois tipos de vértices (pessoas e cargos)
- Um grafo é **bipartido** se
  - $V = X \cup Y; X \cap Y = \emptyset$
  - $\forall (u, v) \in E \; [\, u \in X \wedge v \in Y \,]$

Subgrafos e árvores

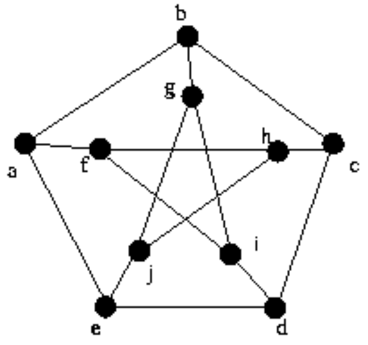
- Um grafo G’ = (V’,E’) é um **subgrafo** de G=(V,E) se
  - $V' \subseteq V \wedge A' \subseteq A$
  - Dado V’ subconjunto de V, o **subgrafo induzido** de G é formado por V’ e todas as arestas incidentes nos vértices de V’
- Uma **árvore** é um grafo conectado sem ciclos
  - Uma árvore pode ou não ter raiz
  - Uma **floresta** é um grafo cujos componentes conexos são árvores
- Uma **árvore geradora** de um grafo conectado é um subgrafo contendo todos os seus vértices e é uma árvore
  - Uma **floresta geradora** de um grafo desconexo é um subgrafo contendo todos os vértices do grafo e cada componente conexo é uma árvore

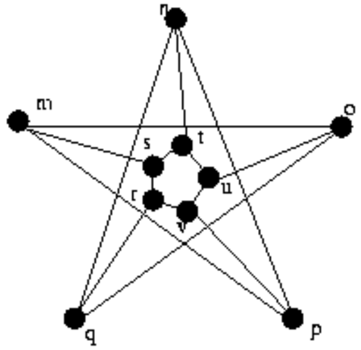
Árvores

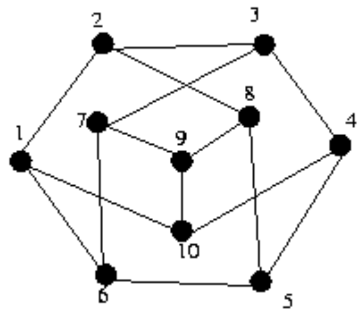
- Seja G=(V,E) um grafo. G é uma árvore se, e somente se, ele satisfaz qualquer das condições a seguir:
  - G tem |V|-1 arestas e não tem ciclos
  - G tem |V|-1 arestas e é conectado
  - G é conectado, mas removendo qualquer aresta o desconecta
  - G é acíclico, mas cria-se um ciclo ao adicionar qualquer aresta
  - Existe somente um caminho entre quaisquer pares de vértices em G

Isomorfismo

- Observe que a definição matemática de grafo não o relaciona diretamente a um desenho
- Dessa forma, dois grafos ‘diferentes’ podem apresentar a mesma estrutura
- Formalmente, para G=(V,E) e G’=(V’,E’), G e G’ são isomorfos (apresentam a mesma estrutura) se existe uma função f: V -> V’ tal que $(u, v) \in E \leftrightarrow (f(u), f(v)) \in E'$







Tipo abstrato de dados Grafo

- Em geral, as seguintes operações estão associadas ao TAD Grafo:
  - Criar grafo vazio com |V| vértices e nenhuma aresta
  - Inserir aresta no grafo
  - Verificar se uma aresta pertence ao grafo
  - Obter lista de vértices adjacentes a um dado vértice
  - Remover aresta do grafo
  - Imprimir grafo
  - Obter número de vértices e arestas
  - Determinar se grafo é direcionado ou não
  - (Em algumas aplicações) obter a aresta de menor custo do grafo

TAD Grafo

- Existem duas formas mais comuns de representar grafos:
  - Matrizes de adjacência
  - Listas de adjacência
- Na primeira, representamos um grafo através de uma matriz |V|x|V| de bits onde cada posição A[i,j] indica se existe aresta entre o vértice i e o vértice j
  - Se G for direcionado, A[i,j] != A[j,i]
  - Se as arestas tiverem pesos, então a matriz deve ser numérica (inteiros ou reais)
  - A complexidade de espaço é $\Omega(|V|^2)$. Podemos armazenar somente a diagonal superior em grafos não direcionados.

TAD Grafo

- A segunda representação, através de listas de adjacência é feita com o auxílio de listas encadeadas
- O grafo é representado por um vetor de tamanho $|V|$ de listas encadeadas
- Cada lista armazena os vértices adjacentes ao vértice da posição da lista no vetor
- Para uma aresta (u,v), v é inserido na lista de adjacentes de u.
  - Se o grafo for não-direcionado, u também é inserido na lista de v
- Alternativamente, podemos usar dois vetores:
  - Um armazena os vizinhos de cada vértice (em ordem)
  - O outro guarda o início das arestas de cada vértice

TAD Grafo

- Complexidade listas de adjacência
  - Espaço: $O(|V| + |E|)$
  - Tempo para verificar aresta: $O(|V|)$
    - $O(\log d(v))$ para implementações usando vetores ordenados
- Complexidade matrizes de adjacência
  - Espaço: $\Omega(|V|^2)$
  - Tempo para verificar aresta: $O(1)$
- Listas de adjacências são úteis para grafos esparsos (poucas arestas)
- Se $|E| = O(|V|^2)$ (grafo próximo a um grafo completo), matrizes de adjacência oferecem melhor desempenho

Leitura

- Seções 4.1 e 4.2 (Sedgewick e Wayne, Algorithms 4ed)
- Seção 3.1 (Kleinberg e Tardos)
- Seções 13.1 (Goodrich e Tamassia)
- Referência da figura:
    - A. Beveridge and J. Shan, "Network of Thrones”, *Math Horizons Magazine* , Vol. 23, No. 4 (2016), pp. 18-22