Max-flow

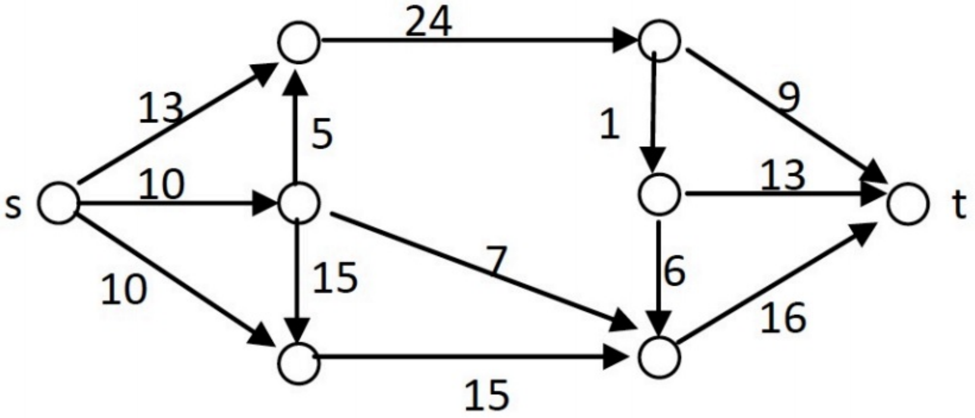
Marco Antonio Gómez Martín

Facultad de Informática - UCM

Max-flow

- Qué es
- Ejemplo
- Algoritmo
- Min-cut
- Min-cost max-flow

Ejemplo de grafo



Aplicaciones de Max-Flow

- Máxima cantidad de datos transmitida por una red
- Max-cardinality bipartite matching
- Asignación de recursos
- ...

Ejemplo: asignación de camisetas

- N personas cada una con distintas preferencias de tallas de camisetas.
- Stock de C_i camisetas de la talla i.

¿Qué talla de camiseta damos a cada persona?

Algoritmo

- Problema muy estudiado.
- Múltiples algoritmos para resolverlo, de distinta complejidad...
- ... y también de distinta complejidad de cara a escribirlos en un concurso.

En la mayoría de los casos, el algoritmo de *Edmonds-Karp* es suficiente.

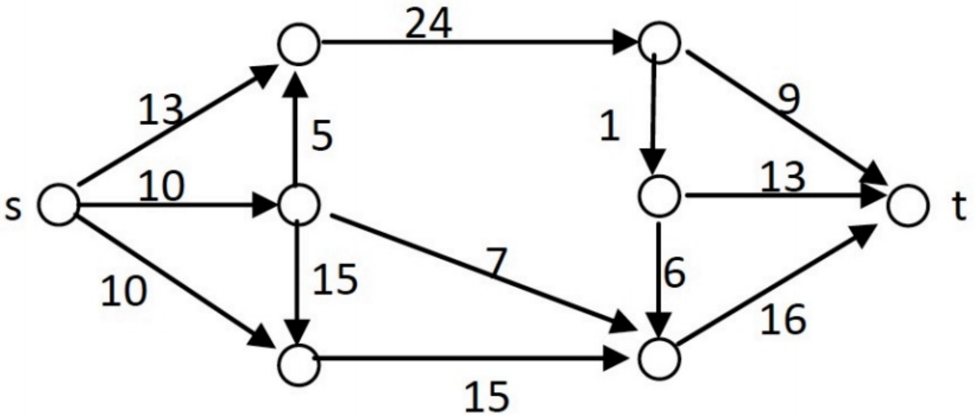
Algoritmo

Idea intuitiva: mandar “flujo” hasta que no se pueda mandar más.

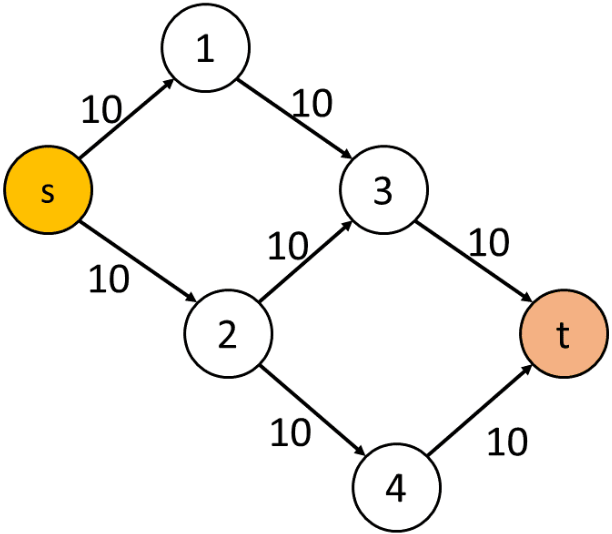
```
int edmondsKarp(int s, int t) {

  int ret = 0;

  int flow = 0;
  do {
   flow = sendFlow(s, t);
   ret += flow;
  } while (flow > 0);

  return ret;
}
```

Algoritmo

- Para comprobar si se puede mandar más flujo, se utiliza BFS.
- Son aristas transitables las que aún tienen capacidad.
- Si se puede ir de s a t, se puede enviar flujo.
- La cantidad de flujo será la capacidad de la arista con menor capacidad.



Ejemplo de ejecución



Algoritmo

- Al mandar flujo, quitamos capacidad a la arista u-v y la añadimos a la arista v-u.
- Eso tiene implicaciones en la implementación.
- En principio, necesaria matriz de adyacencia.
- Se puede usar, además, lista de adyacencia para simplificar el BFS.

Algoritmo

```
// En parent dejamos el anterior en el recorrido BFS
void bfs(int s, int t) {

  queue<int> q;
  memset(visited, 0, sizeof(visited));
  q.push(s);
  parent[s] = -1; visited[s] = true;
  while (!q.empty()) {
   int u = q.front(); q.pop();
   if (u == t) break;
   for (int i = 0; i < adj[u].size(); ++i) {
    int v = adj[u][i];
    if (!visited[v] && (cap[u][v] > 0)) {
      parent[v] = u;
      visited[v] = true;
      q.push(v);
    }
   }
  }
```

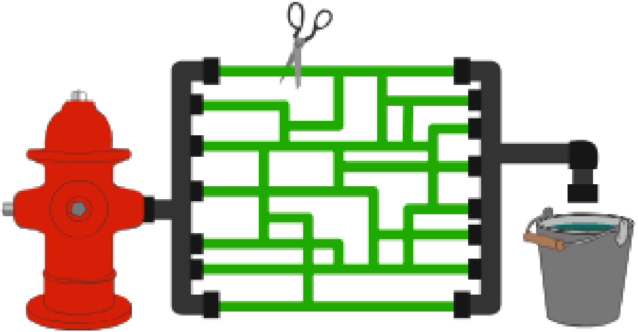
Algoritmo

```
int sendFlow(int s, int t) {
  // Intentamos llegar de s a t
  bfs(s, t);
  if (!visited[t])
   return 0; // No pudimos
  // Buscamos capacidad más pequeña en el camino
  int flow = INF, v = t;
  while (v != s) {
   flow = min(cap[parent[v]][v], flow);
   v = parent[v];
  }
  // Mandamos flujo
  v = t;
  while (v != s) {
   cap[parent[v]][v] -= flow;
   cap[v][parent[v]] += flow; // INVERSA
   v = parent[v];
  }
  return flow;
```

Algoritmo

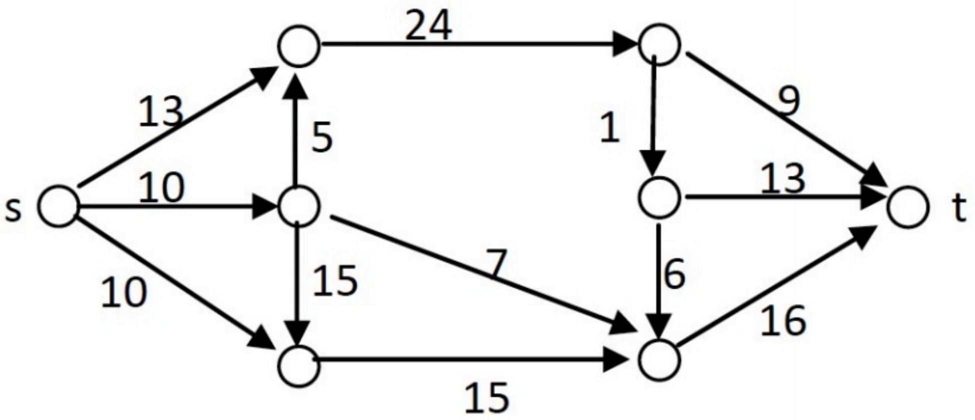
- Complejidad O(VE^2) en tiempo
- Requiere V^2 en memoria para matriz de adyacencia, además de lo necesario para la lista de adyacencia.
- Sin lista de adyacencia la complejidad es O(V^3E)
- Si el grafo es muy grande puede 1) no entrar en memoria y/o 2) dar TLE.

Min-cut



Min-cut

Si las etiquetas son el coste de destruir cada conexión, ¿cuáles cortamos para aislar s y t?



Min-cut

- El coste del min-cut coincide con el valor del max-flow.
- Tras el último BFS, los vértices visitados están en el "lado s". Las aristas que desde esos vértices llegan a vértices no visitados forman parte del min-cut.

Min-cost max-flow

- Las aristas tienen una capacidad y un coste de atravesarlas.
- ¿Cuál es el flujo máximo de mínimo coste?

Algoritmo similar pero en lugar de BFS un algoritmo de búsqueda de caminos mínimos que soporte aristas negativas.

Ejercicio propuesto

820 - Internet bandwitdh