Camino mínimo

Tecnología Digital V: Diseño de Algoritmos
Universidad Torcuato Di Tella

El problema de camino mínimo

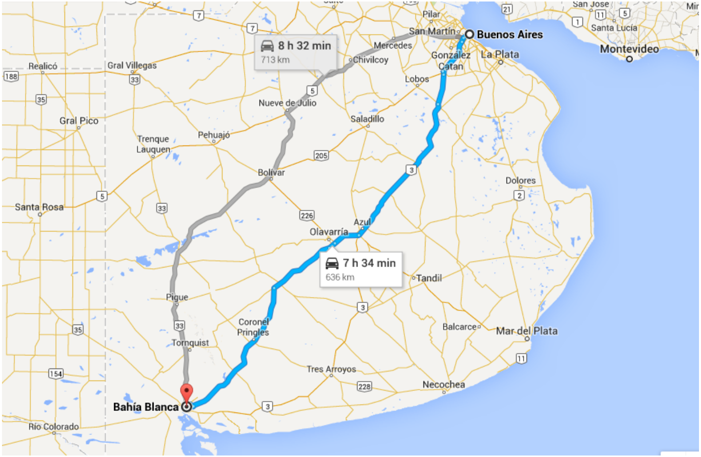
Problema

Dados ...

1. Un grafo dirigido pesado $G = (N, A)$, con una función de distancia $d : A \longrightarrow \mathbb{R}_+$ asociada con los arcos, y
2. nodos $s, t \in N$ de origen y destino,

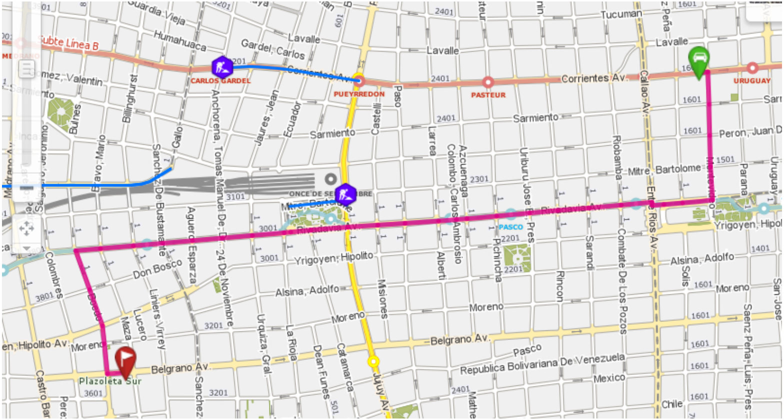
... determinar el camino pesado más corto en G entre los nodos s y t.

Camino mínimo en grafos



Aplicación I

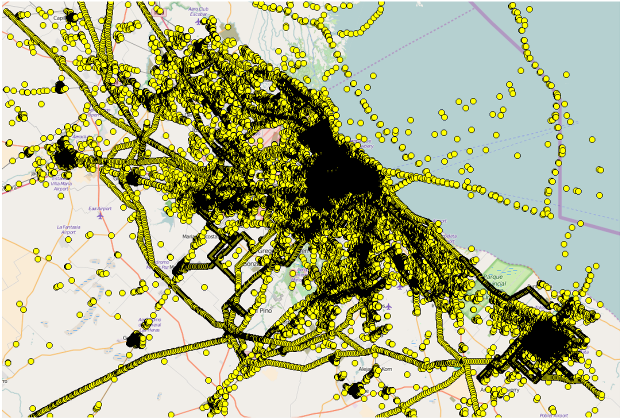
Determinar el camino más corto entre dos ciudades en un mapa vial.



Aplicación II

Determinar el camino más corto entre dos puntos de una ciudad (a pie, en auto, o en bicicleta).

Camino mínimo en grafos



www.openstreetmap.org

- **Arcos con peso negativo:** Si el grafo G no contiene ciclos de peso negativo o contiene alguno pero no es alcanzable desde s, entonces el problema sigue estando bien definido, aunque algunos caminos puedan tener longitud negativa.
- Sin embargo, si G tiene algún ciclo con peso negativo alcanzable desde s, el concepto de camino de peso mínimo deja de estar bien definido.
- **Propiedad de subestructura óptima de un camino mínimo:** Sea P_{ij} un camino mínimo entre i y j, y sea $k \in P_{ij}$. Entonces, el subcamino de P_{ij} entre i y k es un camino mínimo entre i y k.

Algoritmo de Dijkstra (1959)

Edsger Dijkstra (1930–2002)

www.cs.utexas.edu/users/EWD

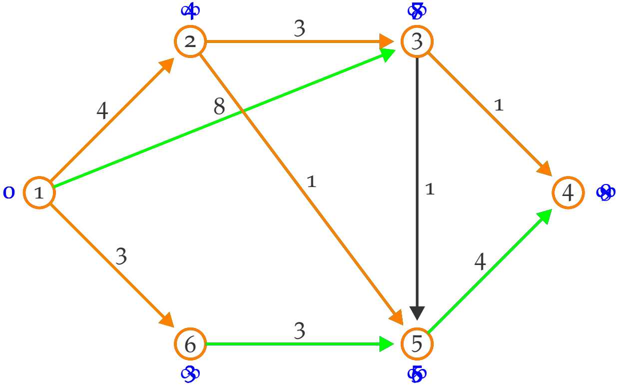
Algoritmo de Dijkstra (1959)

- Asumimos que las longitudes de las aristas son positivas. El grafo puede ser orientado o no orientado.

Algoritmo de Dijkstra

1. Asignar distancias tentativas $d_s = 0$ y $d_i = \infty$ para $i \neq s$.
2. Mientras el destino no esté visitado:
   - Seleccionar como nodo actual i el nodo no visitado con menor distancia tentativa, y marcarlo como visitado.
   - Para cada $j \in N^+(i)$ no visitado, calcular $d'_j = d_i + d_{ij}$. Si $d'_j < d_j$ entonces fijar $d_j := d'_j$.
3. Retornar d_t.

$S = \{1, 6, 2, 5, 3, 4\}$



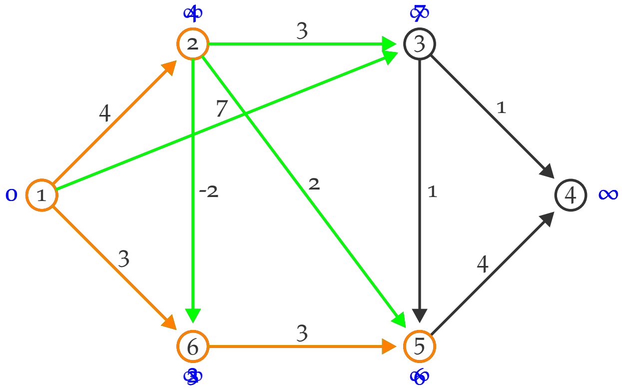
Lema

Llamamos S_k a los nodos visitados luego de la iteración k. Al finalizar la iteración k el algoritmo de Dijkstra determina el camino mínimo entre el nodo s y cada uno de los nodos de S_k.

Teorema

Dado un grafo orientado G con pesos positivos en las aristas, el algoritmo de Dijkstra determina el camino mínimo entre el nodo s y el resto de los nodos de G.

Algoritmo de Dijkstra con pesos negativos



Algoritmo de Dijkstra, complejidad computacional

Algoritmo de Dijkstra

1. Asignar distancias tentativas $d_s = 0$ y $d_i = \infty$ para $i \neq s$.
2. Mientras el destino no esté visitado:
   - Seleccionar como nodo actual i el nodo no visitado con menor distancia tentativa, y marcarlo como visitado.
   - Para cada $j \in N^+(i)$ no visitado, calcular $d'_j = d_i + d_{ij}$. Si $d'_j < d_j$ entonces fijar $d_j := d'_j$.
3. Retornar d_t.

Complejidad computacional:

- $O(n^2)$ con una implementación sencilla.
- $O(m \log n)$ guardando las distancias tentativas en un heap.

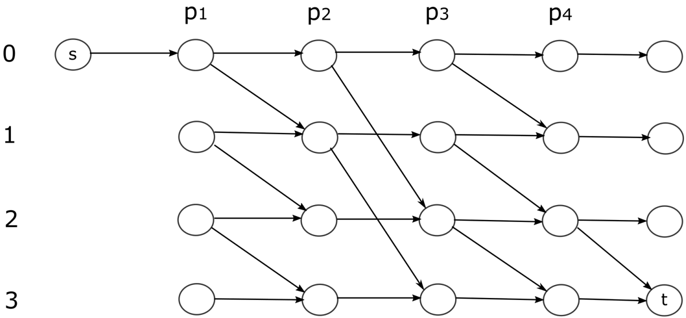
Exact packing

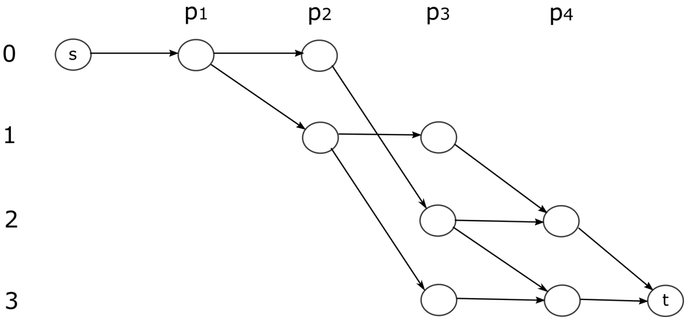
Dados un conjunto $C = \{1, \ldots, n\}$ de objetos, el peso $p_i \in \mathbb{Z}_+$ de cada objeto $i \in C$ y una capacidad máxima $M \in \mathbb{Z}_+$, determinar si existe un subconjunto de C con peso igual a M.

- Ejemplo:

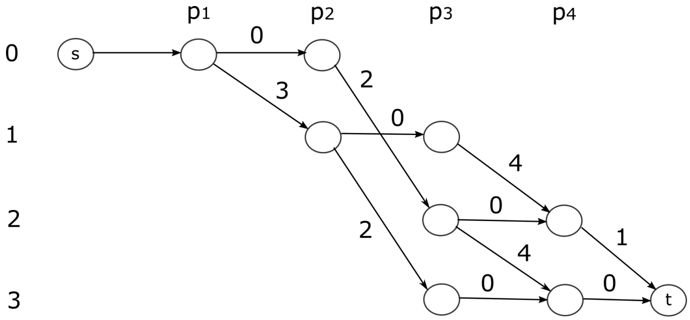
  1. $C = \{1, \ldots, 4\}$,
  2. $p_1 = 1$, $p_2 = 2$, $p_3 = 1$, $p_4 = 1$,
  3. $M = 3$.

- Cómo convertimos este problema en un problema de recorrido en un grafo?


p1
p2
p3
p4
0
1
2
3
s
t


p1
p2
p3
p4
0
1
2
3
s
t

Si además cada objeto tiene un costo de compra y queremos minimizar el costo total ...



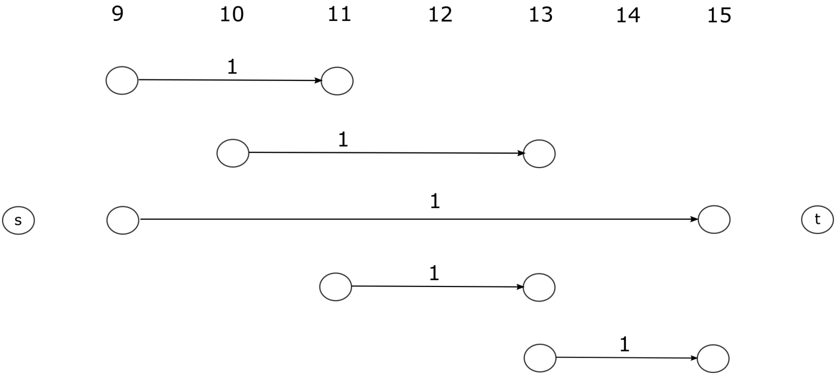
Course scheduling

Dado el programa de charlas de una conferencia, determinar la mayor cantidad de charlas a la que podemos asistir.

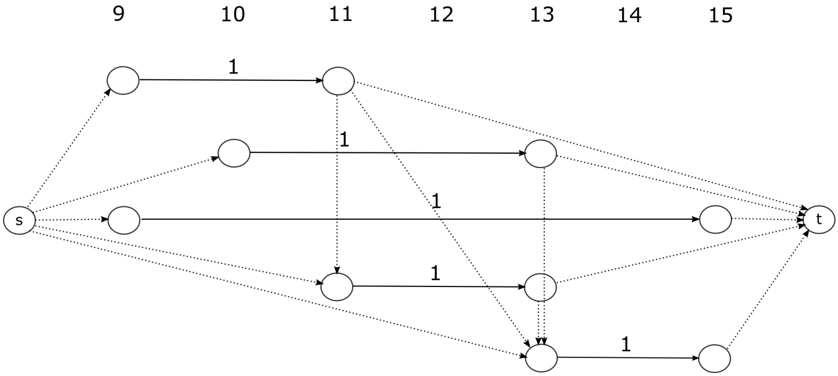
1. Todas las charlas se realizan en un mismo día.
2. Cada charla tiene un horario de inicio y final.
3. Una vez que entramos a una charla, no se puede salir hasta que termine.

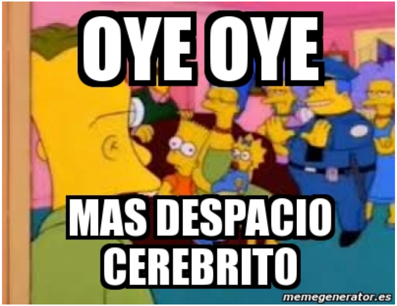
- Cómo convertimos este problema en un problema de camino mínimo?

Ejemplo. [9-11], [10-13], [9-15], [11-13], [13-15]



Ejemplo. [9-11], [10-13], [9-15], [11-13], [13-15]





Camino mínimo con arcos negativos!

1. Algoritmo de Bellman-Ford (1956). Complejidad $O(nm)$ en el peor caso, pero puede manejar arcos con longitudes negativas.
2. Algoritmo de Floyd-Warshall (1962). Complejidad $O(n^3)$. Calcula el camino mínimo entre cada par de nodos, y también permite arcos con longitudes negativas.

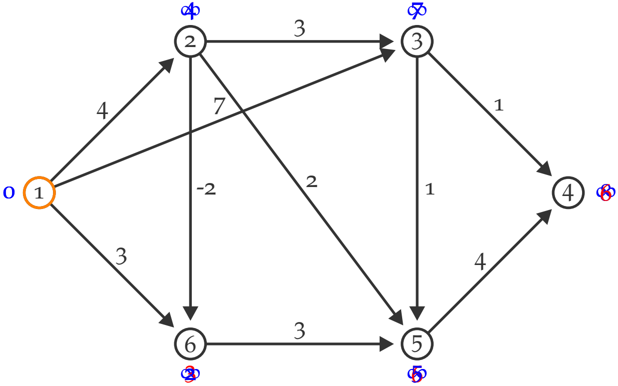
Algoritmo de Bellman-Ford (1955)

- Si G no tiene circuitos de longitud negativa, al finalizar la iteración k el algoritmo de Bellman-Ford determina los caminos mínimos de a lo sumo k arcos entre el vértice s y los demás vértices.
- El grafo puede ser orientado o no orientado. .

Algoritmo de Bellman-Ford

1. Asignar distancias tentativas $d_s = 0$ y $d_i = \infty$ para $i \neq s$.
2. Mientras ocurran actualizaciones:
   - Para cada arista (i, j), si $d_i + d_{ij} < d_j$ entonces fijar $d_j := d_i + d_{ij}$.
3. Retornar d_t.

Algoritmo de Bellman Ford con pesos negativos



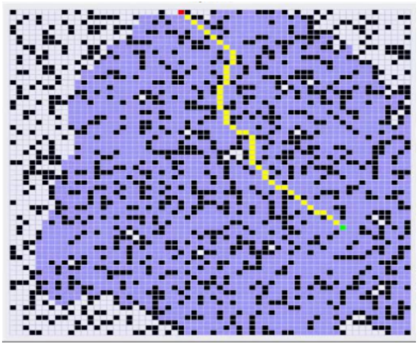
Algoritmo de Floyd-Warshall

- El algoritmo de Floyd-Warshall calcula el camino mínimo entre cada par de nodos de un grafo orientado con pesos en las aristas.
- El invariante del algoritmo es que en la iteración k se determina el camino mínimo entre cada par de nodos i, j que pasa por los nodos $1, \ldots, k$.

Algoritmo de Floyd-Warshall

1. Inicializar $d_{ij} = \infty$ para todo par de nodos $i, j \notin E$.
2. Para cada nodo $k \in N$
3.   Para cada par de nodos $i, j \in N$
4.   Si $d_{ik} + d_{kj} < d_{ij}$ entonces fijar $d_{ij} = d_{ik} + d_{kj}$.

Algoritmo A*



Nodos visitados

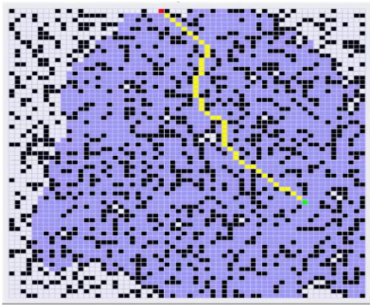
Cuando termina, el Algoritmo de Dijkstra calcula el camino mínimo entre el nodo origen y todos los nodos visitados.

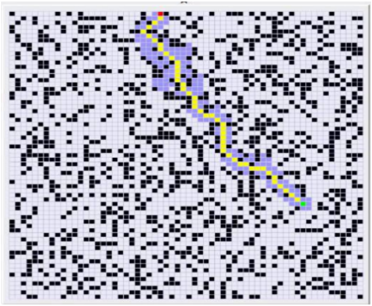
A. Lundblom y R. Uggelberg, *Comparative analysis of weighted pathfinding in realistic environments*. Degree Project in Computer Science. KTH Skolan för Datavetenskap och Kommunikation, Suecia (2017).

Algoritmo A* = Dijkstra + estimación de distancia al destino

- Tenemos como parte de los datos una estimación $\ell : N \longrightarrow \mathbb{R}$ de la distancia entre cada nodo y el nodo destino.
- Suponemos que esta estimación cumple la desigualdad triangular $\ell_i \leq \ell_j + d_{ij}$ para todo arco $ij \in E$.
- En cada paso del Algoritmo de Dijkstra, en lugar de usar la distancia d_{ij} de cada arco $ij \in E$, usamos $\hat{d}_{ij} := d_{ij} + \ell_j - \ell_i$ como su "distancia".
- Si ℓ_i es una cota inferior de la distancia del nodo i al nodo destino, para todo $i \in N$, entonces el Algoritmo A* es un algoritmo. Si no, es una heurística.

Algoritmo A*

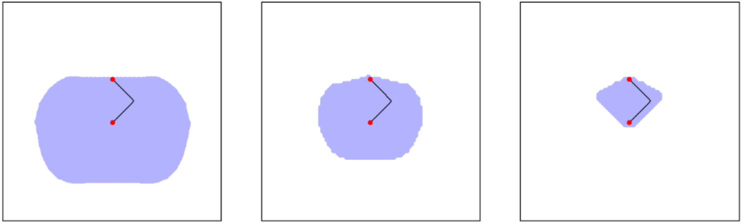




Nodos visitados

- Cuando se utiliza una cota inferior para la función de estimación ℓ, el Algoritmo A* visita una cantidad de nodos mucho menor.
- No obstante, el peor caso sigue siendo el mismo!

A. Lundblom y R. Uggelberg, *Comparative analysis of weighted pathfinding in realistic environments*. Degree Project in Computer Science. KTH Skolan för Datavetenskap och Kommunikation, Suecia (2017).



(a) Algoritmo de Dijkstra (b) A^* con heuristica Cartesiana (c) A^* con heuristica VMC

Nodos visitados

Otro ejemplo, ahora con grafos grilla que representan movimientos de un barco a vela en una regata. El objetivo es llegar en el menor tiempo posible de una boya a la siguiente.

F. Martínez y G. Sainz-Trápaga, *Modelos y algoritmos de optimización combinatoria para planificación de rutas en regatas de barcos de vela*. Tesis de Licenciatura, Universidad de Buenos Aires, Argentina (2010).