Video 41 (1)

Conceptos Principales

1. a
2. b
3. c

Notas

Resumen: Caminos más cortos en grafos valorados (Dijkstra)

La idea general es parecida a cuando buscás caminos más cortos en un grafo normal, pero aquí **cada arista tiene un costo**. Ese costo puede representar cualquier cosa: distancia, tiempo, riesgo, “muertes por zombies”, lo que sea. El punto es que **no todos los caminos valen lo mismo**, entonces no sirve BFS. Ahí entra **Dijkstra**, que es el algoritmo estándar para esto.

Qué hace Dijkstra en palabras simples

Dijkstra busca el camino de menor costo desde un nodo inicial hacia todos los demás, pero lo hace de forma **codiciosa**: siempre elige el siguiente nodo con el costo acumulado más pequeño entre los que aún no están “fijos”. Ese es su truco: avanzar siempre con lo mejor que tiene en el momento.

Funciona así:

1. Empezás en un nodo inicial (A).
2. A sus vecinos les asignás su costo directo (por ejemplo, A → G cuesta 2).
3. A todos los demás les ponés infinito porque todavía no hay forma de llegar.
4. Elegís el nodo con el menor costo actual que aún no está “cerrado”.
5. Desde ese nodo, tratás de mejorar los costos de sus vecinos sumando:

   costo para llegar a ese nodo + peso de la arista hacia el vecino.
6. Si esa suma es menor que lo que ya tenías, lo actualizás.
7. Repetís hasta que todos los nodos estén cerrados.

La diferencia clave con BFS es que aquí **los costos se acumulan**, no los pasos.

Qué estructuras se usan

Dijkstra necesita dos tablas:

1. **Tabla de trabajo (costos):**

   Para cada nodo guardás el costo mínimo conocido hasta ese momento.

   También marcás si el nodo ya está “fijo”.

2. **Tabla de caminos:**

   Para cada nodo guardás desde cuál nodo llegaste con el mejor costo.

   Esto sirve para reconstruir la ruta al final.

En C esto perfectamente se implementa con arreglos paralelos (distancia[], previo[], visitado[]). Nada más exótico.

Qué pasa durante el algoritmo (explicado sin el dibujo del video)

- Comenzás en A con costo 0.
- A sus vecinos (G, D, C…) les asignás sus costos directos.
- De todos esos, G tiene el menor costo (2), entonces se vuelve “fijo”.
- Desde G intentás mejorar los costos de los demás.

  Por ejemplo, si llegar a G cuesta 2 y de G a H cuesta 22, entonces llegar a H costaría 24. Si eso mejora lo que había, lo reemplazás.
- Luego elegís el siguiente nodo más barato entre los que no están cerrados. Sigues repitiendo esto.
- Cada vez que mejorás un costo, actualizás también quién es el “padre” de ese nodo.
- Esto va avanzando hasta que todos los nodos quedan con el menor costo posible.

El video sigue con el mismo grafo del vídeo de BFS, pero ahora los valores cambian porque ya no se trata de conteo de pasos sino de sumar costos.

Lo importante que sí tenés que entender para exámenes

- **Dijkstra solo funciona cuando no hay pesos negativos.**
- Siempre elegís el nodo sin cerrar con la distancia mínima actual.
- Cuando fijás un nodo, su costo queda definitivo (igual que en BFS, pero ahora es por costo).
- Para cada vecino, revisás:

  costo actual del nodo + peso de la arista.

  Si es mejor, actualizás.
- La tabla de “previos” es lo que permite reconstruir el camino real después.
- Concepto clave: **algoritmo codicioso** → siempre tomás la mejor opción disponible en ese momento.

Por qué esto sirve muchísimo en programación en C

- C te obliga a implementar las tablas a mano, así que entendés bien qué está pasando.
- Muchos ejercicios piden simular el algoritmo paso por paso.
- Es típico en estructuras de datos porque combina grafos, arreglos y lógica algorítmica clara.
- Entender Dijkstra te ayuda a entender otros algoritmos más avanzados de grafos.