Examen validación ADA 2019-2020

1) En el algoritmo de Dijkstra con cola se guarda en la cola…
    a. Aristas.
    b. Vértices.
    c. Parejas de vértices y su distancia actual.
    d. Distancias.
    e. Ninguna de las anteriores.

2) ¿Qué algoritmos se han tenido que implementar en la pràctica?
   -2 Dijkstra, 1 Greedy, 2 Backtracking y 3 Branch and Bound.

3) La clase CGraph guarda...
    a. list <CVertex> m_Vertices y list<CEdge> m_Edges.

4) La clase CVertex guarda...
    a. El punto del plano donde está el vértice.
    b. La lista de los vértices vecinos.(año pasado tenía vértices ahora tiene los edges vecinos así que damos por buena la d)
    c. La distancia calculada por el algoritmo de Dijkstra.
    d. a,b y c.
    e. Ninguna de las anteriores.

.

5) Errata, en el examen la pregunta 4 y 5 eran la misma.

6) En la pràctica 3 (Backtracking)...
   b. El camino resultante de un vértice a utilizar a otro vértice a visitar, pasa como mucho una vez por cada vértice del grafo.

7) Una cola con prioridad se declara en C++ como:
    a. Priority_queue<tipo de los elementos de la cola, tipo de contenedor, función de comparación> queue;

8) ¿Cuáles son las extensiones de los ficheros con que trabaja el programa salesman?
   -.GR para grafos, .DIS para las distancias, .VIS para visitas.

9) Si tenemos una lista de visitas con más de 2 vértices que empiezan y acaban en el mismo vértice… (NO sabemos si esta bien, pero la c y d de base son descartables, porque dijkstra no utiliza visitas, y el backtracking si está bien implementado debería de poder lidiar con esta problemática. La a es como muy redundante y la b también porque no hay forma de asegurar que dicho grafo tenga solución.)
    a. Solo existe un camino óptimo.
    b. Existen dos o más caminos óptimos.
    c. El algoritmo de Dijsktra se queda en bucle infinito.
    d. El algoritmo de backtracking se queda en bucle infinito.
    e. Ninguna de las anteriores.

10) ¿Cuál es el algoritmo más lento de los implementados?
    -Backtracking puro.

Examen recuperación validación ADA 2019-2020

1) Un CGraph com representa un graf?

a) Llista de vertex i cada vertex guarda la llista de veïns.
b) Una matriu adjacència on cada element és la dist de vèrtex.
c) Una llista d'arestes.
d) Una llista de vertex i una llista d'arests. Para este año.

2)Una llista amb prioritat es declara a c++ com?

a) priority_queue<tipus dels elements, vector<tipus element>, comparador>.queue;

3)Quins algoritmes hem programat?

-Djikstra, Djisktra queue, greedy, backtracking pur, backtracking-greedy, Branch&bound 1, Branch&bound 2, Branch&bound 3.

4)A la practica un objecte CVertex..

a) Nomes pot existir dintre de la classe CGraph. En altres casos será un apuntador o referencia a CVertex.
b) Nomes pot existir dintre de la classe CVisits i CTrack. En altres casos será una apuntador o referencia a un CVertex.
c) És convenient copiar els objectes CVertex per poder treballar amb ells.
d) És convenient crear nous objectes CVertex en lloc d'utilitzar apuntadors a CVertex.
e) Cap de las anteriors.

5)Els objects CTrack..

a) Conté una llista de punts en el pla que representen els vertex.
b) No se li pot associar un graf i ja conté els seus propis vertex .
c) Es poden concatenar.
d) Totes les anteriors.
e) Cap de les anteriors.

6)Els elements de la cua amb prioritat utilitzada per implementar l'algoritme de Djikstra amb cua són…

a) CVertex.
b) *CVertex.
c) Parelles de CVertex i distancia actual.
d) CEdje*.
e) Cap de les anteriors.

7)Quin és el número mínim de nodes de l'algoritme de Djikstra que la de fer algoritme greedy per solucionar el problema del viatjant de comerç.

a) n*n vertex com és el número de vertes a visitar.

b) n*(n-1) vertex, on n es el número de vertex a visitar.
c) n nodes on n és el número de vertex a visitar.
d) n-1 nodes on n és el número de vertex a visitar.
e) Cap de les anteriors.

8)Com poden accedir a tots el veïns d'ún vertex?

a) for (CVertex *pVei: v.m_neighbours;) processar pVei; Este año seria con CEdge en vez de CVertex: for (CEdge* e : actual->m_Edges)
b) for (CVertex vei: v.m_neighbours;) processar pVei;
c) for (CVertex &vei: v.m_neighbours) processar vei;
d) for (CVertex &vei: v.getNeighbours()) processar vei;
e) Cap de les anteriors.

9)Quin és el mínim numero d'operacions necesaries per calcular la nova cota inferior d'una solució fila a partir de una cota inferiror de la solució amb l'heuristica 2 de branch & bound.

NOTA: dudamos entre 2.

a) Restar la long del camí més llarg per arribar al vèrtex que afegim.
b) Tantes operacions com vertex tingui la solució mare.
c) Sumar la longitud del camí que porta des del final del camí de la solució mare al vèrtex que afegirem.
d) Tantes sumes com vèrtex a visitar tingui la solució mare menys 1.
e) Cap de les anteriors.

10)Els algoritmes de backtracking implementats en la practica…

NOTA: No sabemos si esta bien.

a) El backtracking pur i el backtracking greedy seleccionen els vertex del graf per on pasa el cami.
b) El backtracking greedy és més lent que el pur.
c) El backtracking pur selecciona les visites del graf que formen el camí i el greedy selecciona l'arbre de visita de C?
d) El backtracking pur utilitzar com a base els camins de l'algoritme de Djikstra.