Algoritmos y Estructuras de Datos II - 15 de junio de 2016

Segundo Parcial

Alumno: .....

Siempre se debe explicar la solución, una respuesta correcta no es suficiente sino viene acompañada de una justificación que demuestre que la misma ha sido comprendida. Las explicaciones deben ser completas. La utilización de código c o de nivel de abstracción excesivamente bajo influirá negativamente. **Por favor, resolvé cada ejercicio en una hoja aparte,** para acelerar el proceso de corrección. **¡No olvides escribir claramente tu nombre en cada una de las hojas!**

1. A continuación se encuentra el algoritmo de Dijkstra y una variante del mismo:

```
fun dijkstra(L: array[V,V] of costo, v: V)
                                  ret D: array[V] of costo
    var x: V
    C:= V-{v}
    for j ∈ V do D[j]:= L[v,j] od
    do C ≠ ∅ → x:= elemento de C con D[x] mínimo
               C:= C-{x}
               for j in C do
                    D[j]:= min(D[j],D[x]+L[x,j])
               od
    od
end fun
```

```
fun dijkstra2(L: array[V,V] of costo, v,w: V)
                                  ret D: array[V] of costo
    var x: V
    C:= V-{v,w}
    for j ∈ V do D[j]:= min(L[v,j],L[w,j]) od
    do C ≠ ∅ → x:= elemento de C con D[x] mínimo
               C:= C-{x}
               for j in C do
                    D[j]:= min(D[j],D[x]+L[x,j])
               od
    od
end fun
```

Para el siguiente grafo,

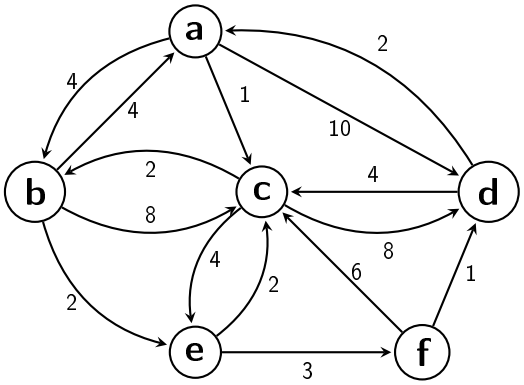
(a) Dar la matriz de adyacencia L.

(b) Ejecutar paso a paso dijkstra(L,b), graficando el vector D, luego de cada ejecución completa de un ciclo **for**.

(c) Ejecutar paso a paso dijkstra2(L,b,f), graficando el vector D, luego de cada ejecución completa de un ciclo **for**.

(d) Describir con la mayor precisión posible qué hace en general (para todo grafo G) el algoritmo dijkstra.

(e) Describir con la mayor precisión posible qué hace en general (para todo grafo G) el algoritmo dijkstra2.



2. Dados los valores naturales n y k, y la siguiente definición recursiva:

$$m(i,j) = \begin{cases} i & \text{si } j = 0 \\ m(0,j-1) + m(1,j-1) & \text{si } i = 0 \wedge j > 0 \\ m(n-1,j-1) + m(n,j-1) & \text{si } i = n \wedge j > 0 \\ m(i-1,j-1) + m(i,j-1) + m(i+1,j-1) & \text{si } i > 0 \wedge i < n \wedge j > 0 \end{cases}$$

para $i \in \{0,1,\ldots,n\}$ y $j \in \{0,1,\ldots,k\}$, escribir un algoritmo que utilice programación dinámica para calcular $m(n,k)$ a partir de los argumentos n y k.

Sugerencia: comenzar graficando una matriz de $n+1$ filas y $k+1$ columnas (para n y k pequeños) e intentar calcular (algunos de) los valores a mano.

3. Se dispone de una balanza de dos platillos (llamémosles A y B) con un objeto sobre el platillo A cuyo peso es W. Se disponde de n pesas, cuyos pesos son $w_1, w_2, \ldots, w_n$. Se pide dar un algoritmo que utilice backtracking para determinar cuál es el menor número de pesas que alcancen para equilibrar la balanza. Es muy importante tener presente que pueden colocarse pesas en ambos platillos. Por ejemplo, si se cuenta con 4 pesas de 1 ($w_1 = w_2 = w_3 = w_4 = 1$) y con una pesa de 5 ($w_5 = 5$), y $W = 2$ el menor número de pesas es 2, lo que se logra colocando dos pesas de 1 en el platillo B. Si $W = 4$, el menor número sigue siendo 2, ya que podemos equilibrar la balanza colocando una pesa de 1 en el platillo A y la de 5 en el platillo B. Si $W = 3$ hay dos formas de obtener el menor número de pesas, que es 3.

   Para obtener el algoritmo se sugiere definir $m(i, j)$ = "el menor número de pesas, eligiendo entre las primeras i pesas, que alcancen para equilibrar la balanza cuando el peso en el platillo A excede en j al peso del platillo B". Con esta definición, el resultado deseado se obtiene calculando $m(n, W)$. Observar que para el caso ilustrado en los ejemplos de arriba ($w_1 = w_2 = w_3 = w_4 = 1$ y $w_5 = 5$) se obtiene $m(5, 2) = m(5, 4) = 2$ y $m(5, 3) = 3$.

   (a) Inciso de entrenamiento. Para el ejemplo dado ($w_1 = w_2 = w_3 = w_4 = 1$ y $w_5 = 5$) ¿cuánto valen $m(5, 1)$, $m(5, 5)$, $m(5, 0)$, $m(2, 1)$, $m(1, 2)$, $m(2, -1)$, $m(5, -5)$? En los dos últimos, j asume valores negativos, lo que significa que el platillo A tiene menos peso que el B.

   (b) Dar una definición recursiva de $m(i, j)$, para el caso general, es decir, cualquiera sean $w_1, w_2, \ldots, w_n$ y W. Puede convenir analizar los siguientes casos: ¿qué ocurre con la balanza cuando j es 0? ¿qué ocurre si i es 0 y $j \neq 0$? En el caso general, $i > 0$ y $j \neq 0$, ¿qué opciones tengo con la i-ésima pesa?

4. A continuación se encuentran los algoritmos iterativos para recorrer un grafo en DFS y en BFS respectivamente.

```
proc dfsearch(in G, in/out mark: tmark, in v: V)
  var p: stack of V
  empty(p)
  visit(mark,v)
  push(v,p)
  while ¬is_empty(p) do
    if existe w ∈ neighbours(top(p)) tq ¬visited(mark,w)
    then visit(mark,w)
         push(w,p)
    else pop(p)
    fi
  od
end
```

```
proc bfsearch(in G, in/out mark: tmark, in v: V)
  var q: queue of V
  empty(q)
  visit(mark,v)
  enqueue(q,v)
  while ¬is_empty(q) do
    if existe w ∈ neighbours(first(q)) tq ¬visited(mark,w)
    then visit(mark,w)
         enqueue(q,w)
    else dequeue(q)
    fi
  od
end
```

   Para el grafo del primer ejercicio (ignorando los costos de las aristas),

   (a) Ejecutar paso a paso dfsearch(G,mark,e), graficando la pila luego de cada modificación de la misma. Asignar los números que correspondan a cada vértice según el orden en que fueron visitados.

   (b) Ejecutar paso a paso bfsearch(G,mark,e), graficando la cola luego de cada modificación de la misma. Asignar los números que correspondan a cada vértice según el orden en que fueron visitados.