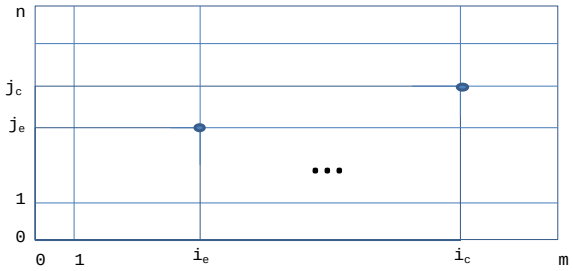
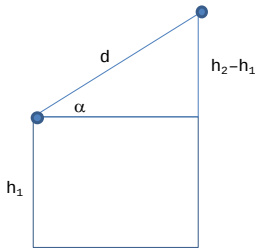
ISIS 2103 Diseño de algoritmos **Nombre:** _____
Semestre 2011-10 Parcial 3 **Sección:** _____
Mayo 12, 2011
Prof. Rodrigo Cardoso

1 **[40 puntos]** Las ciclovías de una ciudad están definidos por una red de *calles* (caminos sur-norte) y *carreras* (caminos este-oeste). Las calles son todas paralelas entre sí y, así mismo, las carreras. La nomenclatura de la ciudad numera las calles de `0` a `m`, de forma consecutiva, y las carreras, de `0` a `n`, con $m,n \geq 2$. La *esquina* `(i,j)` es el cruce de la calle `i` con la carrera `j`. Una *cuadra* es un trayecto de calle o carrera que conecta dos esquinas consecutivas (en el orden de la calle o carrera del trayecto). Las cuadras se denotan citando el par de esquinas que unen.





La alcaldía ha levantado planos de la ciudad en los que se anotan las distancias en metros para cada cuadra. Estas son `d(i,j,i+1,j)` y `d(i,j,i,j+1)` siempre que los índices estén en rango.

También se anotan las alturas sobre el nivel del mar, en metros, de cada una de las esquinas. Estas son `h(i,j)`, para índices en rango.

La casa de Juan está en la esquina (i_c,j_c), y él estudia en la esquina (i_e,j_e). Como ciclista aficionado, Juan prefiere las rutas con cuadras menos pendientes, no importa la distancia que deba recorrer. Obsérvese en la figura de la derecha que la pendiente de una cuadra, entendida como la tangente del ángulo α, es calculable con los datos de los planos de la alcaldía. Así, si la pendiente es positiva en un sentido de la cuadra, es negativa en el otro sentido.

1a **[30/40]** Use una versión del algoritmo de Dijkstra para encontrar el camino que involucra cuadras menos pendientes de la casa de Juan a su sitio de estudio.

Justifique su respuesta (identifique semianillos, grafos, costos y explique la aplicabilidad de soluciones conocidas).
(N.B.: Se califican mejor los algoritmos más eficientes)

El grafo

Considérese el grafo dirigido $G(V,\rightarrow,p)$, donde

$V=0..m \times 0..n$: esquinas **[5/30]**

$.\rightarrow.$: cuadra orientada en V, i.e., $(a,b)\rightarrow(c,d)$ entre esquinas consecutivas. **[5/30]**

$p(a,b,c,d)$ $= (h(c,d)-h(a,b))/\sqrt{(d(a,b,c,d)^2 - (h(c,d)-h(a,b))^2)}$
: pendiente de la cuadra $\langle(a,b),(c,d)\rangle$. **[5/30]**

Un camino en `G` es una sucesión de cuadras orientadas que conectan esquinas.

El costo de un camino es la máxima pendiente involucrada en él. Se quiere calcular el camino de menor costo entre los puntos (i_c,j_c) y (i_e,j_e).

Se usará una versión del algoritmo de Dijkstra sobre un semianillo apropiado.

El seminanillo

Sea $\mathbf{R}^{\infty} = \mathbf{R} \cup \{\infty, -\infty\}$.

- $(\mathbf{R}^{\infty}, \min, \infty)$ es monoide conmutativo
- $(\mathbf{R}^{\infty}, \max, -\infty)$ es monoide (conmutativo).
- $a \max(b \min c) = (a \max b) \min (a \max c)$
- $a \max \infty = \infty$

Entonces, $(\mathbf{R}^{\infty}, \min, \max, \infty, -\infty)$ es un semianillo.

[10/30]

El algoritmo

Se propone un algoritmo de Dijkstra en $(\mathbf{R}^{\infty}, \min, \max, \infty, -\infty)$, con parada forzada si se alcanza el nodo objetivo.

[5/30]

1b [10/40] Estime las complejidades temporal y espacial de su algoritmo.

Complejidad espacial:

Un vector `d[0..m*n-1]` adicional.

$S(m,n) = \theta(mn)$

[2/10]

Complejidad temporal:

En el cuerpo del algoritmo debe considerarse el cálculo de pendientes de nodos conectados con los recién visitados. Sin embargo, este cálculo es $O(1)$ y no afecta la complejidad temporal.

El número de arcos es $e = 2mn$. El número de vértices es mn. Entonces, $e = 2mn \leq mn \log mn$ si y solo si $2 \leq \log mn$, i.e., $4 \leq mn$ (lo cual es cierto, ya que $m,n \geq 2$). Por lo tanto, se puede tener una implementación que encuentre soluciones en tiempo

$T(m,n) = \theta(mn \log mn)$

[8/10]

Variante penalizada 1

No analizar número de arcos, pero dar respuesta $\theta(mn \log mn)$

[4/10]

Variante penalizada 2

Versión cuadrática de Dijkstra

$T(m,n) = \theta(m^2n^2)$

[4/10]

Análisis más fino

Cada esquina está conectada con, a lo sumo, otras 4 esquinas. De este modo, el paso de relajación del algoritmo de Dijkstra cuesta $O(1)$ y se hace una vez por cada esquina visitada. Es decir, en realidad: $T(m,n) = \theta(mn)$

[+10/20]

2 **[40/40]** Una empresa tiene un conjunto de puntos de venta de sus productos. Hay conexiones directas entre algunos de los puntos y, para cada par de puntos conectados directamente, se conoce el valor del costo de transportar un kilo de uno al otro. Se sabe que hay conexión directa de cada punto a, por lo menos, la mitad de los demás.

Se quiere ubicar una bodega central en uno de los puntos, de manera que el promedio aritmético del costo de transportar un kilo desde la bodega hasta los puntos de venta sea mínimo. ¿Dónde debe situarse la bodega?

2a [24/40] Describa un algoritmo para solucionar el problema.

Se puede modelar el problema con un grafo $G(V,E,c)$, donde:

V : conjunto de puntos de venta, digamos, $1..n$
E : arcos entre puntos. El arco $(i,j) \in E$ ssi hay posibilidad de transporte de productos de i a j.
c : costo de transportar 1 kilo de i a j, si $(i,j) \in E$.

$G(V,E,c)$ se representa con una matriz de distancias $D \in M_n(\mathbf{R}^*)$, tal que:

$D[i,j] = c(i,j)$, si $(i,j) \in E$
$= \infty$, en otro caso

La bodega debe quedar situada en un punto x tal que

$$(+j|\ 1 \le j \le n : D^*[x,j]/n) \le (+j|\ 1 \le j \le n : D^*[i,j]/n) \quad \text{, para } 1 \le i \le n$$
$$\equiv$$
$$(+j|\ 1 \le j \le n : D^*[x,j]) \le (+j|\ 1 \le j \le n : D^*[i,j]) \quad \text{, para } 1 \le i \le n$$
$$\equiv$$
$$x = (\min i|\ 1 \le i \le n : (+j|\ 1 \le j \le n : D^*[i,j])) \quad \text{(C)}$$

[8/24]

Algoritmo:

1 Determinar D^*
Variante 1: Floyd-Warshall
Variante 2: Dijkstra n veces.

[8/24]

2 Encontrar x tal que (C) .
Minimizar sobre las sumas de filas de D^*

[8/24]

2b [16/40] Estime las complejidades temporal y espacial de su algoritmo.

La Parte 1 del algoritmo debe costar
$T1(n) = O(n^3)$

No importa qué variante: Dijkstra debe gastar también $T1(n) = O(n^3)$, porque $e \geq n*n/2 > n \log n$.
En espacio, también cualquier variante usará

$S1(n) = O(1)$ (calculando la matriz D^* en la matriz D)

[4/16]

La Parte 2 cuesta:

$T2(n) = O(n) * O(n)$ // Calcular n sumas, de n sumandos y minimizar
$= O(n^2)$
$S2(n) = O(1)$

[4/16]

Resumen:

$T(n) = T1(n) + T2(n)$
$= O(n^3) + O(n^2)$
$= O(n^3)$

[4/16]

$S(n) = S1(n) + S2(n)$
$= O(1) + O(1)$
$= O(1)$

[4/16]

3 **[20 puntos]** Dados dos números naturales $m,n>0$, considere el problema de construir una secuencia de números s tal que:

$s.0 = 0$	// el primer elemento de s es 0
$s.i < m,\ 0 \leq i < \#s$	// todos los elementos son menores que m
$s.i < s(i+1),\ 0 \leq i < \#s-1$	// la secuencia es creciente
$(+i\mid 0 \leq i < \#s : s.i) = n$	// la suma de los elementos de la secuencia es n

3a [10/20] Exprese el problema como una búsqueda en grafos.

$SOLPOS = \{s:Seq\ \mathbf{nat} \mid s \text{ creciente}, s.0=0, (+i \mid 0 \leq i < \#s : s.i) \leq n\}$

[3/10]

$sat.s \equiv (+i \mid 0 \leq i < \#s : s.i) = n$

[2/10]

$BUSQ = SOLPOS$

[2/10]

$\langle s.0, s.1, \ldots, s.i \rangle \rightarrow \langle s.0, s.1, \ldots, s.i, k \rangle$, si $s.i < k < m$

[2/10]

$s = \langle 0 \rangle$

[1/10]

3b [10/20] Justifique si (i) hay que marcar nodos (ii) hay que verificar que la agenda se vacíe. (iii) el algoritmo puede no terminar.

(i) No hay que marcar. El grafo no tiene ciclos.

[2/10]

(ii) Sí: puede no haber solución por agenda vacía.

Ejemplo: $m=1$, $n=1$. La única secuencia en BUSQ es $\langle 0 \rangle$. La agenda se vacía en el primer ciclo.

[4/10]

(iii) No: el número de nodos en BUSQ es finito. Debe terminar.

Cada nodo tiene ramificación finita. Si no terminara debería haber un camino de longitud arbitrariamente grande. Si se toma un camino de longitud mayor que n, los elementos de la última secuencia de este camino sumarían más de n, y ésta no pertenecería a BUSQ. Por esto, BUSQ debe ser finito.

[4/10]

Variante

Ninguna secuencia puede medir más de `m` (y el máximo número que se puede construir es `m(m-1)/2`).

[4/10]