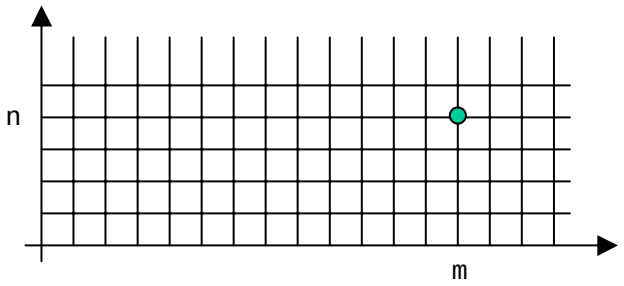
ISIS 211 Diseño de Algoritmos
Semestre 2003-1 - Parcial 2
Abril 3, 2003
Prof. Rodrigo Cardoso

---

1 Un agente secreto se encuentra en el origen (posición $\langle 0,0 \rangle$) de una ciudad cuadriculada y desconocida, cuyo plano tiene la forma:



La ciudad tiene muchas calles y carreras ("muchas" ≈ infinitas!). El agente sabe que su contacto se encuentra en una cierta posición $\langle m,n \rangle$. Aunque ignora las coordenadas exactas, su contacto le ha comunicado que desde el origen a su posición hay exactamente r maneras de llegar desde el origen, caminando siempre hacia la izquierda o hacia arriba, y que, además, $m \geq n > 1$. Por ejemplo, si el contacto está en $\langle 3,2 \rangle$, le ha comunicado que está en un punto al que se puede llegar de 10 maneras.

a **[40 puntos]** Utilice programación dinámica para explicar cómo puede el agente averiguar las coordenadas exactas de su contacto, i.e., $\langle m,n \rangle$.

(definición de lenguaje, recurrencia, diagrama de necesidades, invariante). No es necesario que escriba su algoritmo.

b **[20 puntos]** Estime las complejidades espacial y temporal de su algoritmo en términos de m y n. Para lo temporal, use la asignación como operación básica.

---

2 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 puede quedar la bodega?

a **[24 puntos]** Describa un algoritmo para solucionar el problema

b **[16 puntos]** Estime las complejidades temporal y espacial de su algoritmo.

ISIS 211 Diseño de Algoritmos
Semestre 2003-1 – Solución Parcial 2
Abril 3, 2003
Prof. Rodrigo Cardoso

---

1 [60/60]

1a [40/40]

Lenguaje

`nc(i,j) = "maneras de ir desde el origen a ⟨i,j⟩", para i≥j≥0`

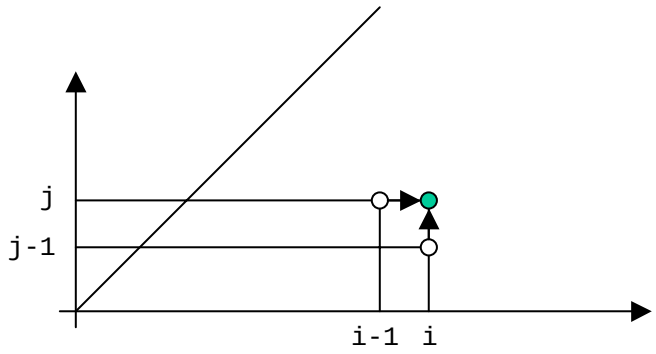
[8/40]

Recurrencia:

```
nc(i,j)  =   1                        , si i≥j=0
         =   nc(i-1,j) + nc(i,j-1)    , si i≥j>0
```

[12/40]

Diagrama de necesidades:

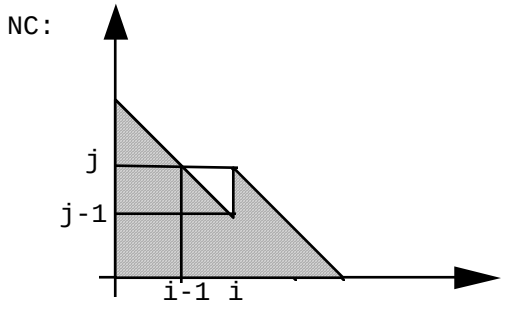


El diagrama sugiere un recorrido del cuadrante por diagonales de pendiente `1` que se alejan del origen. Las diagonales pueden ser recorridas ascedentemente, i.e, empezando la diagonal `i`-sima en `⟨i,0⟩`. En algún momento se tendrá `nc(i,j)=r`, para ciertos `i≥j>0`, según lo prometido.

Entonces: `⟨m,n⟩ = ⟨i,j⟩`.

[10/40]

Invariante: (se requiere una estructura de datos `NC`, no acotada, pero finita)



[10/40]

1b [20/20]

Complejidades:

La posición del contacto se encuentra en la diagonal `m+n`. La diagonal `i` tiene `i+1` puntos en los que debe ser calculada la función `nc`.

$$S(m,n) = O(1+2+...+(m+n)) = O((m+n)^2/2) = O(m^2)$$

[5/20]

Algo mejor: bastan dos vectores que guarde las dos últimas diagonales. Así:

$$S(m,n) = O(m+n) = O(m)$$

[10/20]

El tiempo es proporcional al área barrida:

$$T(m,n) = O((m+n)^2/2) = O(m^2)$$

[10/20]

Hay mejoras posibles. Observando que $nc(x,y)=nc(y,x)$, la mitad superior de cada diagonal puede determinarse conociendo la mitad inferior. Más aun: no debería calcularse esa parte superior, porque la respuesta está en la parte inferior de la diagonal. Sin embargo, debe calcularse hasta la diagonal. Puede aprovecharse el hecho de que

$$nc(i,i) = nc(i-1,i)+nc(i,i-1) = 2\ nc(i,i-1).$$

Variaciones

El cuadrante se puede recorrer, por ejemplo, con cuadrados de lado $1, 2, \ldots$, cuya esquina inferior izquierda esté en el origen. Para calcular un cuadrado a partir del anterior se calculan los lados izquierdo (excepto la esquina derecha) y superior. En este caso el contacto se encuentra en el cuadrado de lado $\max(m,n)=m$. Las complejidades son $O(m^2)$. La espacial puede mejorarse, guardando los lados derecho y superior del cuadrado, i.e., $O(m+m) = O(m)$.

Como en la primera solución, se puede aprovechar la simetría de `nc` para reducir a la mitad los cálculos necesarios.

---

2 [40/40]

2a [24/24]

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 $(i,j)\in E$ sii 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 $D\in \mathbb{R}^{n}_{*}$, tal que:

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

La bodega debe quedar situada en un punto x tal que

$(+j|\ 1\le j\le n\ :\ c^*[x,j]/n) \le (+j|\ 1\le j\le n\ :\ c^*[i,j]/n)$, para $1\le i\le n$

≡

$(+j|\ 1\le j\le n\ :\ c^*[x,j]) \le (+j|\ 1\le j\le n\ :\ c^*[i,j])$, para $1\le i\le n$ (C)

[8/24]

Algoritmo:

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

[8/24]

2 Encontrar x tal que (C) (puede describirse; no es necesario el código)

```
{Pre: D = c*}
u:= 0;
for j:= 1 to n → u:= u+D[1,j] rof;
x,k:= 1,1;

{Inv: 1≤k≤n+1 ∧ u=(+j|1≤j≤n:c*[x,j]) ∧ (∀i|1≤i<k:u ≤(+j|1≤j≤n:c*[i,j]))}

do k≠n+1 → v:= 0;
           for j:= 1 to n → v:= v+D[k,j] rof;
           if v<u then v:= u fi;
           k:= k+1
od
{Pos: u=(+j|1≤j≤n:c*[x,j]) ∧ (∀i|1≤j≤n:u ≤(+j|1≤j≤n:c*[i,j]))}
```

[8/24]

2b [16/16]

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 $e \ge n*n/2 > n \log n$.
En espacio, también cualquier variante usará

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

La Parte 2 cuesta:

$T2(n) = O(n^2)$
$S2(n) = O(1)$

Resumen:

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

[8/16]

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

[8/16]