Ayudantía 17

Repaso Examen

Ejercicio 1
Repaso I1

Las claves en un árbol binario de búsqueda están ordenadas. Por lo tanto, debería ser fácil encontrar la k-ésima clave más pequeña o determinar cuántas están en el rango [a, b].
Describe en pseudocódigo cómo podrías encontrar:

a) La k-ésima clave
b) La cantidad de claves dentro del rango [a,b] *(puede suponer que tanto a como b están en el árbol)*
Considera que los nodos podrían llegar a necesitar almacenar información adicional.

2018-2 Ex-P1



Pregunta 1. a) Ex-2018-2

Podemos guardar en cada nodo el tamaño del sub árbol que tiene al nodo como raíz, es decir:
size = # nodos en subárbol izq + # nodos en subárbol der + 1

Ahora, con eso en consideración podemos plantear lo siguiente:

```
k-esima(x,k):
  ranking = x.izq.size + 1 # Es la posición relativa del subárbol
  if k == ranking:
    return x
  elif k < r: # Es decir, buscamos un elemento menor
    return k-esima(x.izq, k)
  else:
    return k-esima(x.der, k-r) # k-r es k relativo al subarbol der
```

Pregunta 1. b) Ex-2018-2

Si queremos saber cuántos elementos hay entre a y b basta con saber qué posición ocupan si los elementos estuviesen ordenados y restar la posición de b con a.

```
posición(x):
  posicion_relativa = nodo.izq.tamaño + 1 # relativa en su subárbol
  actual = x
  while actual no sea raíz:
    if actual es hijo derecho:
      posicion_relativa += actual.padre.izq.size + 1
    actual = actual.padre
  return posicion_relativa
```

En particular, la cantidad de elementos en el rango [a, b] sería:
posición(b) - posición(a) + 1
#Esto porque por ejemplo 7-4=3, pero sería 4,5,6,7: 3+1.

Ejercicio 2
Repaso I2

En el puerto de San Antonio, una empresa logística prepara el plan de embarque de contenedores, en el buque “EDD”, hacia distintos destinos. El buque tiene una capacidad máxima de carga W (toneladas). Cada contenedor i tiene las siguientes características:

- Peso total (toneladas): w_i
- Valor comercial de la carga (USD): v_i
- Destino: puerto p_i, determina restricciones de seguridad para el contenedor.

Pregunta 3 - I2-2025-2

Durante la planificación, se deben seleccionar los contenedores que serán embarcados de modo que:

i. No se exceda la capacidad total W del buque.
ii. Se maximice el valor comercial total de la carga transportada.
iii. No se pueden fraccionar contenedores (no es posible embarcar parcialmente uno).

Pregunta 3 - I2-2025-2

El jefe de operaciones propone usar un enfoque **codicioso (greedy)**:

> "Ordenemos los contenedores según su relación valor/peso (v_i/w_i), de mayor a menor, y carguemos tantos como quepan, en ese orden.".

Este enfoque prioriza la eficiencia económica por unidad de peso.

Responda:

a) (1 pt.) Pruebe, mediante un contraejemplo, que la estrategia codiciosa propuesta no entrega necesariamente la carga óptima.

Pregunta 3 - I2-2025-2

Pregunta 3.a) - I2-2025-2

Propondremos un contraejemplo específico:



Tenemos un buque con capacidad máxima de carga $W=10$

Y tenemos 3 contenedores disponibles para cargar:

1

$w_1=6$

$v_1=100$

2

$w_2=5$

$v_2=60$

3

$w_3=5$

$v_3=60$

Pregunta 3.a) - I2-2025-2

Bajo la estrategia Greedy propuesta, cada bloque tendría el siguiente rate:

1

$w_1=6$

$v_1=100$

$v/w=100/6\approx16.6$

2

$w_2=5$

$v_2=60$

$v/w=60/5=12$

3

$w_3=5$

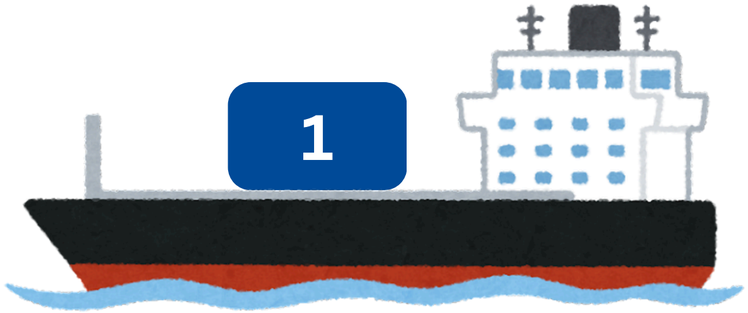
$v_3=60$

$v/w=60/5=12$

La estrategia nos dice que debería cargar el contenedor 1, pero este ocupa un peso de 6, y como el resto de los contenedores tienen peso 5, no puedo cargar ningún otro contenedor. Esto nos lleva a un valor no óptimo!

Pregunta 3.a) - I2-2025-2

Si cargo solo el contenedor 1,



Mi peso utilizado es de 6, y el valor entregado total es de 100

En cambio, si cargo el contenedor 2,



Tengo peso utilizado de 5 y valor 60. Aun me sobra 5, así que añado el contenedor 3. Así, w=10 y valor=120

b) (1 pt.) Modele el problema como un **CSP** (problema de satisfacción de restricciones) a resolver utilizando **programación dinámica**. Identifique las variables, sus dominios, las restricciones y la función a optimizar.

Pregunta 3 - I2-2025-2

Pregunta 3.b) - I2-2025-2

El primer paso es preguntarnos que queremos usar en la funcion objetivo y restricciones. Nos interesa cargar el mayor valor posible, cumpliendo con el peso. Ambas cosas son **propiedades de los contenedores**, así que estos pasarán a ser nuestra variable.

Diremos que un contenedor es c_i y lo utilizaremos como booleano de si agregamos o no un contenedor al buque. Si lo cargamos, su valor será 1 (true). Sino, será 0.

Así,

- **Variables**: c_i por cada contenedor
- **Dominio**: c_i es 1 o 0.

Pregunta 3.b) - I2-2025-2

Ahora que tenemos la variable a usar y su dominio, definiremos entonces las restricciones y la función objetivo.
La **única restricción** es que el peso máximo de carga del buque no se sobrepase. Entonces, la manera matemática de plantear esto es:

$$\sum w_i \times c_i \leqslant W$$

La **función objetivo** es maximizar el valor acumulado del buque. Dado que nuestra variable es booleana, simplemente podemos decir:

$$max\left(\sum v_i \times c_i\right)$$

c) (3 pts.) Describa como se estructuran los subproblemas a resolver y generalice este subproblema para el paso k de resolución del problema. Proponga una recurrencia para resolver el problema y el pseudo código de un algoritmo a partir de ella.

Pregunta 3 - I2-2025-2

Pregunta 3.b) - I2-2025-2

La solución óptima va a ser un conjunto de contenedores {c_1, c_2, ...,c_n}. Estos son de tipo binario, es decir, si el valor de un contenedor es 1, esto quiere decir que lo cargamos al buque y que **pertenece al óptimo**. Podemos estructurar esta decisión como nuestro subproblema, de manera que:

$$g_0(W) = max\{\underbrace{g_1(W - w_0) + v_0}_{\text{Incluyo c_1}}, \overbrace{g_1(W)}^{\text{No incluyo c_1}}\}$$

Pregunta 3.b) - I2-2025-2

Bajo el subproblema que planteamos, para la ecuación de recurrencia basta con generalizar :D
Podemos plantearla de la siguiente manera:

No incluyo c_k

$$g_k(w) = max\{\underbrace{g_{k+1}(w - w_k) + v_k}_{\text{Incluyo c_k}}, \overbrace{g_{k+1}(w)}\}$$

En donde,

$g_n(w) = 0$ si w>=0
$g_n(w) = -\infty$ si w<0

Pregunta 3.b) - I2-2025-2

Luego, plantear el pseudocódigo se da en base a la ecuación de recurrencia.

```
opt(j,n,w):
1 if j = n :
2     if w < 0 : return -∞
3     else : return 0
4 else:
5     if M[j] ≠ ∅ :
6           return M[j]
7     else:
8           M[j]← max{v_{j+1} + opt(j+1,n,w-w_{j+1}), opt(j+1,n,w)}
9           return M[j]
```

Con llamada inicial opt(0,n,W), con W el peso máximo de carga del buque.

OJO: la ecuación de recurrencia de la pauta de esta prueba es de indices 1-based mientras que la nuestra usa 0-based. Por eso varía la ecuación. :)

d) (1 pt.) Considere que, por restricciones de seguridad, existen pares de contenedores (r,s) que no pueden viajar juntos en el buque. Describa brevemente cómo incluiría esta restricción en su solución anterior.

Pregunta 3 - I2-2025-2

Pregunta 3.b) - I2-2025-2

Lo que queremos al implementar esto es que si tenemos un par de contenedores (r,s) que no pueden viajar juntos, es que si r=1, entonces automáticamente s=0. Esto cuenta como una nueva restricción! Pero dada su naturaleza, es suficiente usar condicionales:

```
…
if j = s && r = 1 : return -∞
…
if j = r - 1 :
  r = 1
  con_r = opt(j+1,n,w-w_{j+1})
  r = 0
  sin_r = opt(j+1,n,w)
  M[j]← max{v_{j+1} + con_r, sin_r}
…
```

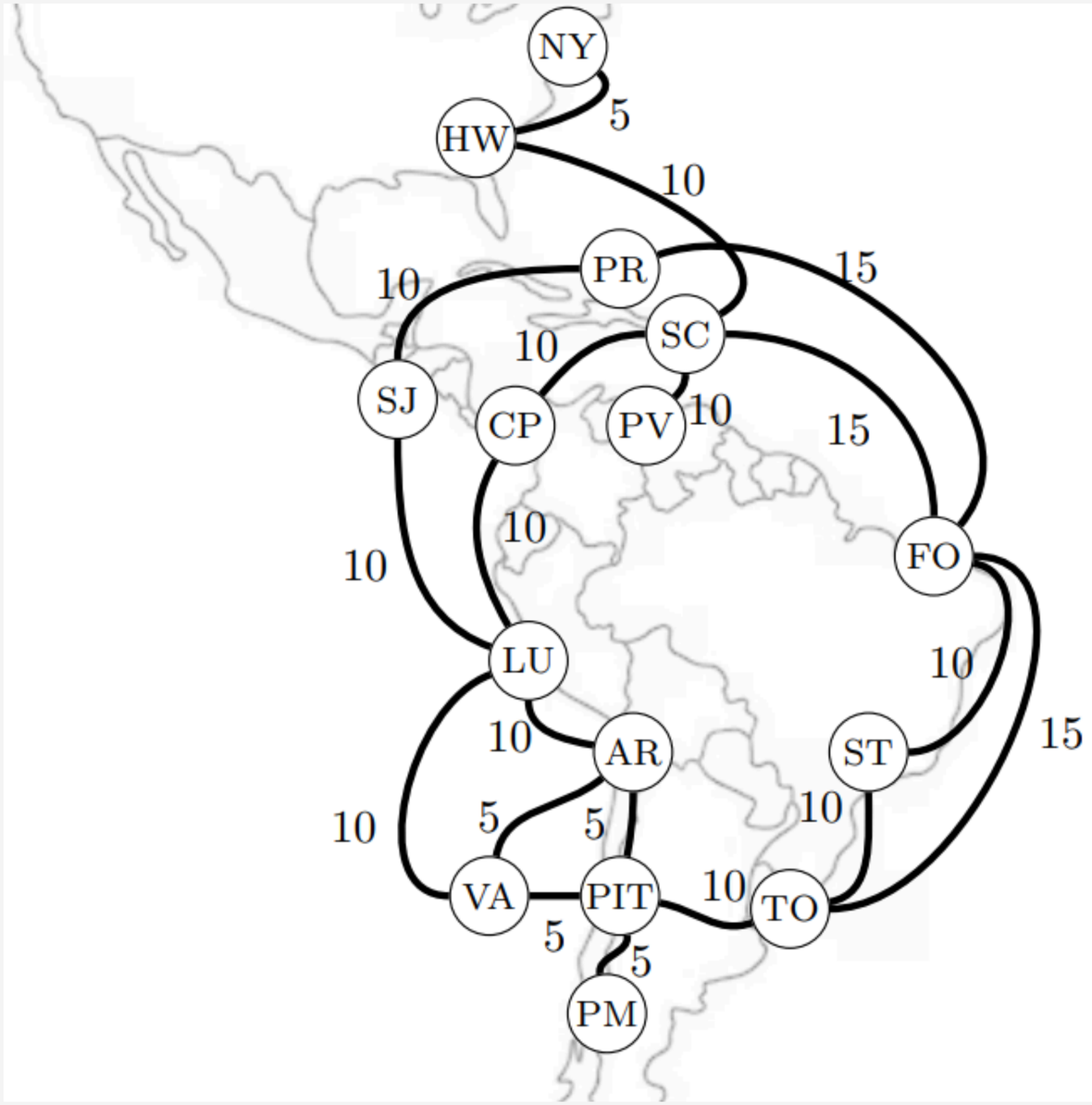
En este caso, la pauta incorpora r y s como argumentos de opt.

OJO: se siguen usando indices 1-based acá. Varía un poco según nuestra implementación :D

Ejercicio 3
Grafos

La red de fibra óptica terrestre y submarina que conecta nuestros computadores puede verse como una grafo no dirigido. En este contexto, se pueden utilizar los protocolos OSPF y RIP para determinar las rutas a utilizar. Considere el siguiente fragmento de la red, donde se indican las latencias como costos en aristas.

Pregunta 4 - I3-2023-2



NY	Brookhaven, NY, EEUU
HW	Hollywood, FL, EEUU
PR	San Juan, PR, EEUU
SC	St. Croix, Virgin Islands, EEUU
PV	Puerto Viejo, Venezuela
CP	Colón, Panamá
SJ	Puerto San José, Guatemala
FO	Fortaleza, Brazil
ST	Santos, Brazil
LU	Lurín, Perú
TO	Las Toninas, Argentina
AR	Arica, Chile
VA	Valparaíso, Chile
PM	Puerto Montt, Chile
PIT	Santiago, Chile

(a) [1 pto.] Los protocolos de ruteo OSPF y RIP utilizan los algoritmos de Dijkstra y Bellman-Ford, respectivamente. Indique si es necesario realizar alguna modificación al grafo mostrado para poder aplicar estos protocolos. Justifique brevemente.

Pregunta 4.a - I3-2023-2

Pregunta 4.a - I3-2023-2

El grafo mostrado es no dirigido, pero tanto Dijkstra como Bellman-Ford operan sobre grafos dirigidos. La modificación necesaria es reemplazar cada arista no dirigida con costo c por dos aristas dirigidas con el mismo costo c. Esto modela que las conexiones de fibra óptica son bidireccionales.

Sobre el grafo dirigido resultante, ambos algoritmos pueden aplicarse sin inconveniente adicional, ya que todos los pesos son positivos, lo cual satisface el requisito de Dijkstra, y Bellman-Ford no impone restricciones sobre el signo de los pesos.

(b) [3 ptos.] Utilizando el algoritmo de Dijkstra, determine la ruta más barata desde Santiago hasta Lurín. Para esto, indique el estado del arreglo de costos d y ancestros π al término de cada iteración, hasta obtener el óptimo pedido. Puede usar las etiquetas de los nodos como índices para simplificar su trabajo.

Pregunta 4.b - I3-2023-2

Pregunta 4.b - I3-2023-2

Se ejecuta Dijkstra desde PIT:

- para cada vecino v del nodo extraído u,
  si $d[u]+c(u,v)<d[v]$, se actualiza $d[v]\leftarrow d[u]+c(u,v)$ y
  $\pi[v]\leftarrow u$.

- Dijkstra garantiza que al momento de extraer un nodo, su distancia es óptima.

Pregunta 4.b - I3-2023-2

Indicamos en la siguiente tabla los pares d[u], π[u] para cada nodo u. Las columnas representan iteraciones y las celdas vacías representan (∅, ∞). Las celdas con -- indican que el nodo ya se extrajo de la cola de prioridad y sabemos que ya se tiene su óptimo.

	iteración					
nodo	0	1	2	3	4	5
PIT	∅, 0	--	--	--	--	--
AR		PIT, 5	--	--	--	--
VA		PIT, 5	PIT, 5	--	--	--
TO		PIT, 10	PIT, 10	PIT, 10	PIT, 10	--
PM		PIT, 5	PIT, 5	PIT, 5	--	--
LU			AR, 15	AR, 15	AR, 15	AR, 15
ST						TO, 20
FO						TO, 25

Pregunta 4.b - I3-2023-2

Dado que Dijkstra conoce el costo óptimo al extraer cada nodo de la cola, sabemos que el óptimo para ir de PIT a LU tiene costo 15 y corresponde al camino

$$\text{PIT}, \ldots, \pi[\pi[\text{LU}]], \pi[\text{LU}], \text{LU}$$

es decir PIT, AR, LU.

(c) [2 ptos.] Considere que se quiere privilegiar el uso de las aristas (LU,CP) y (SJ,PR) cambiando su peso por -30 y -25 respectivamente. Determine si con este cambio es posible usar el algoritmo de Bellman-Ford para determinar la ruta más barata de Santiago a Lurín. Justifique su respuesta.

Pregunta 4.c - I3-2023-2

Pregunta 4.c - I3-2023-2

Al modificar las aristas indicadas, existe un ciclo de costo negativo dado por

LU, CP, SC, FO, PR, SJ, LU

que es alcanzable desde PIT. El costo total de este ciclo es -5 < 0.
Bellman-Ford no puede determinar una ruta de costo mínimo desde PIT hasta LU en este caso. La razón es que al existir un ciclo negativo alcanzable desde la fuente, siempre es posible reducir el costo de cualquier camino que pase por ese ciclo dando una vuelta adicional, por lo que no existe un camino de costo mínimo bien definido.

Pregunta 4.c - I3-2023-2

Cabe destacar que Bellman-Ford sí es capaz de detectar la existencia de este ciclo negativo mediante su pasada de verificación final:

Si tras las $|V| - 1$ iteraciones alguna arista todavía puede relajarse, el algoritmo reporta que hay un ciclo negativo (return false), pero no puede entregar una ruta óptima.

Feedback