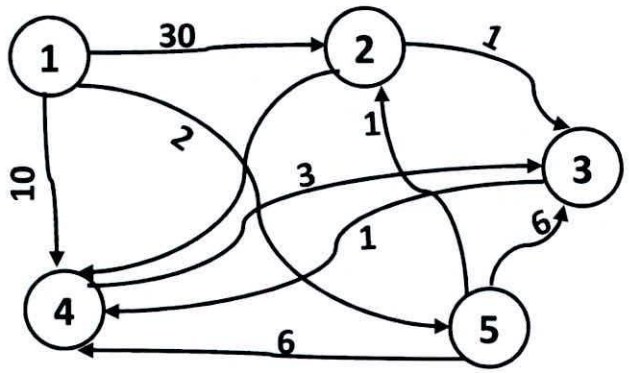
Algoritmos y Programación II, cátedra Calvo.

Febrero 2015

1. Para el siguiente grafo, obtenga los valores correspondientes a los caminos mínimos partiendo del nodo 1. Describa el algoritmo usado en forma precisa. Explique cuál fue la estrategia usada, en qué se basa la misma y en qué tipos de problemas suele aplicarse. ¿Cuál es el coste temporal del algoritmo usado según su implementación?



2. Desarrolle un algoritmo que permita obtener el número de niveles de un árbol binario de búsqueda (no se pide codificar, pero puede hacerlo si lo desea). Desarrolle un algoritmo que permita determinar si un ABB está balanceado por su altura con una diferencia permitida 1. (no se pide codificar, pero puede hacerlo si lo desea)

3. Explique detalladamente en qué consiste la estrategia Divide y Vencerás. Describa un algoritmo en el cual se aplique esta estrategia mostrando en qué parte se lleva a cabo cada etapa, e indique cuál es el coste temporal del algoritmo elegido. Indique, si existe, la relación entre la estrategia Divide y Vencerás y la Modularización

4. Defina cola con prioridades. Describa dos implementaciones distintas, y de diferente coste temporal. Explique cuál es el coste temporal de cada implementación, justificando sus respuestas.

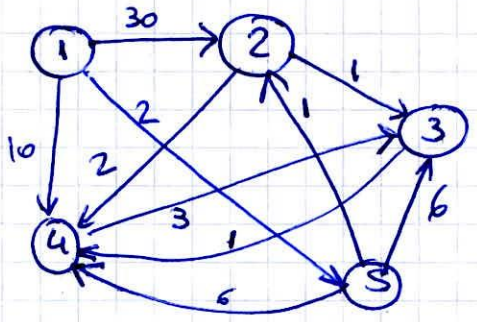
5. Defina hashing y escriba dos formas diferentes de resolver colisiones.

Para aprobar es necesario tener correctos y completos el 60% de cada ítem propuesto.
En la nota se ponderan, también, los resultados de los parciales y trabajos prácticos.
Duración del examen: 2 horas.

HOJA Nº

FECHA Feb - 15

① Para el sig. grafo obtenga los valores correspondientes a los caminos mínimos partiendo del nodo 1. Describa el algoritmo usado en forma precisa. Explique cuál fue la estrategia usada, en qué se basa la misma y en qué tipo de problema suele aplicarse.
¿Cuál es el costo temporal del algoritmo usado según su implementación?



Dijkstra: en este algoritmo se utiliza la estrategia Greedy.

(GPS, mínimo costo viaje, mínimo tiempo).

Dijkstra:

- Armo M → matriz con valores aristas de entrada de un vértice a otro → $O(V^2)$ V: # de vértices
- Creo lista con vértices y los marco como no visitados, menos el inicial → nodo 1 → $O(V-1)$
- Armo vector D en donde voy guardando los pesos mínimos "actualizados" → inicializo s/ Matriz

Mientras ∃ v no visitado (no cuento al 1) → V-1 iteraciones

{ elijo v con menor peso en D y lo marco como visitado.
∀ u adyacente a v

{ D[u] = min (D[u], D[v] + M[v][u]); }

}

?

Costo

$(V+A)+\log(V+A)$

vértice arista

M	1	2	3	4	5
1	0	30	∞	10	2
2	∞	0	1	2	∞
3	∞	∞	0	1	∞
4	∞	∞	3	0	∞
5	∞	1	6	6	0

D	2	3	4	5
desde 1	30	∞	10	2
	3	8	8	2
	3	4	5	2
	3	4	5	2
	2	4	5	2

v = 5 → u(5) = 3, u(5) = 2, u(5) = 4

v = 2 → u(2) = 3, u(2) = 4

v = 3 → u(3) = 4

v = 4 → u(4) = 3

D(u)	D(v) + M(v u)
∞	2 + 6 = (8)
30	2 + 1 = (3)
10	2 + 6 = (8)
8	3 + 1 = (4)
8	3 + 2 = (5)
5	4 + 1 = 5
4	5 + → 4

Visitados	2	3	4	5
	↑	↑	↑	↑

desde 1 a

2	3	4	5
2	4	5	2

② Desarrolle un algoritmo que permita obtener el número de niveles de un árbol binario de búsqueda (no se pide codificar). Desarrolle un algoritmo que permita determinar si un ABB está balanceado por su altura con dif. permitida 1.

```
int alturaMax (pNodo* punt) {
    si (punt != NULL) {
        Si (punt -> der == NULL) && (punt -> izq == NULL)
            { return 0; }
        si no
            return 1+ max(alturaMax (punt -> der), alturaMax(punt -> izq));
        }
    }

bool estaBalanceado (pNodo* punt) {
    return ((abs (alturaMax (punt -> der) - alturaMax (punt -> izq)) < 2));
    }
```

NOTA

HOJA Nº

FECHA

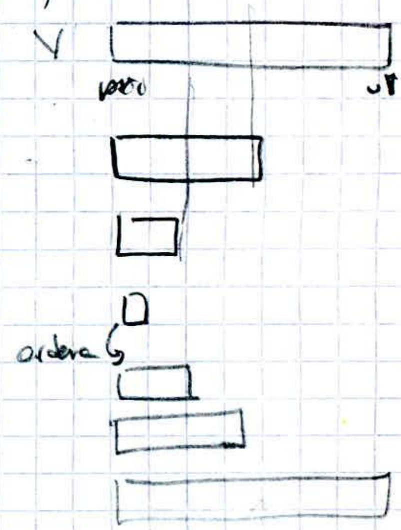
(3) Explique detalladamente en qué consiste la estrategia D y V. Describa un algoritmo en el cual se aplique esta estrategia mostrando en qué parte se lleva a cabo cada etapa e indique cuál es el coste temporal del algoritmo elegido. Indique, si ∃ la relación entre estrategia D y V y la modularización.

Dado un problema de tamaño N < L (arbitrario), se divide en subproblemas con la misma estructura que el problema original, y son independientes entre sí.
Para los problemas de tamaño menor que L se resuelve de otra manera.
La solución general es una combinación de las sol. parciales. (Subproblemas)

Un algoritmo en donde se emplea esta estrategia es en el método de ordenamiento Merge sort.

```
void MS (int V[], int pri, int ult) {
O(1)      Si (pri < ult) {
O(1)          int medio = (pri + ult) / 2;
T(n/2)        MS (V, pri, medio);
T(n/2)        MS (V, medio+1, ult);
(n)           Intercalar (V, pri, ult);
          }
}
```

} divide al problema en 2 partes (en forma recursiva



$T(n) = 2\,T\left(\frac{n}{2}\right) + n$ → x teo Maestro ($a = 2$, $b = 2$, $k = 1$)

$T(n) \in O(n \log(n))$

NOTA

④ Defina cola con prioridades. Describa dos implementaciones ≠ y de diferente coste temporal. Explique cuál es el coste temporal de %empl.

Es una estructura en donde se guardan datos acolandolos, pero teniendo en cuenta una prioridad.

todos los elementos de mayor prioridad se encuentran adelante de todos aquellos que tienen menor prioridad.

Implementaciones: → Heap de máxima (en donde se guarda por prioridad) → al principio O(n) o de mínimo O(log(n))

→ AVL → pero para empezar a armarlo O(n log(n))

⑤ Defina hashing y describa 2 formas ≠ de resolver colisiones

Es una alternativa a los algoritmos de búsquedas basados en comparación de claves. Consiste en aplicar una función (hash) a una clave y transformarla en un valor. Sea una transformación de la clave

Cuando al aplicar hash a 2 claves ≠ da el mismo valor se produce colisión → $h(x_1) = h(x_2)$ → colisión

Se puede resolver en forma abierta o cerrada

dentro de Direcc. abierto al tener una colisión se puede ubicar el dato en el espacio siguiente de memoria (vector) o aplicar una nueva función hash (rehashear) para ubicarlo en otro lugar