



UNIVERSIDAD
DE GRANADA

Universidad de Granada
Departamento de Ciencias de la Computación e Inteligencia Artificial
E.T.S.I. Informática y de Telecomunicación, C/Periodista Daniel Saucedo Aranda s/n - 18015 - Granada (España)

Grado en Ingeniería Informática Algorítmica

Curso 2021/2022. Convocatoria ordinaria.

23 de junio de 2022

1. (2 puntos) Resolver la ecuación en recurrencias $T(n) = 3T(n/3) + \log_3(n)$, indicando el orden resultante asumiendo que la ecuación se obtuvo para el caso peor de un algoritmo.
2. (2 puntos) Una empresa de gestión de aguas ha encontrado un nuevo pozo capaz de abastecer a una serie de N pueblos de una comarca. Para diseñar la estrategia de abastecimiento deciden calcular el coste de interconectar (mediante tuberías) cada par de pueblos x e y entre sí, $c(x, y)$, y cada pueblo directamente con el pozo p , $c(x, p)$. Diseñad un algoritmo para que dicha empresa pueda abastecer de agua a los N pueblos usando ese pozo de manera que el coste de construir toda la infraestructura sea el mínimo. El algoritmo debe encontrar la solución óptima y además hacerlo de forma eficiente.
(Algoritmo visual)
3. (2 puntos) Sea T un conjunto de n programas, cada uno con un tamaño t_1, t_2, \dots, t_n (los números t_i son enteros positivos), y disponemos de un dispositivo de almacenamiento de capacidad máxima C (también un número entero positivo). Se pretende encontrar la manera de decidir qué programas (completos) almacenar en el dispositivo de manera que se maximice el espacio ocupado del dispositivo. Diseñad un algoritmo de programación dinámica para resolver este problema. Aplicadlo para el caso en que $n = 4$, el vector de tamaños es $t(1) = 3, t(2) = 7, t(3) = 11$ y $t(4) = 8$, con capacidad del dispositivo $C = 12$.
(Algoritmo de la mochila)
4. (2 puntos) Se desea conocer si, dado un número natural n , éste es o no un número combinatorio del tipo $\binom{k}{2}$, para algún número k . Por ejemplo, $36 = \binom{9}{2}$, o $499500 = \binom{1000}{2}$. Especificad un algoritmo lo más eficiente posible para resolver este problema. Nota: $\binom{k}{2} = \frac{k*(k-1)}{2}$.
(DyU zu = k(k-1) (k = sqrt(zu))
5. (2 puntos) Dados n números positivos w_i , $1 \leq i \leq n$, y un entero más M , se trata de encontrar todos los subconjuntos de números w_i cuya suma valga exactamente M . Por ejemplo, si $n = 4$, ($w_1 = 11, w_2 = 13, w_3 = 24, w_4 = 7$) y $M = 31$, entonces los subconjuntos buscados son $(11, 13, 7)$ y $(24, 7)$. Se pide realizar un algoritmo de exploración en grafos para solucionar el problema.
(Búsqueda en grafos)

Duración del examen: 2 horas y 30 minutos.

NOTA: En todos los problemas se debe emplear alguna de las técnicas de diseño o análisis de algoritmos estudiadas en esta asignatura.

$$1. \quad T(u) = 3T\left(\frac{u}{3}\right) + \log_3(u)$$

Para resolverla, aplicamos el siguiente cambio de variable

$$u = 3^w$$

obteniendo

$$T(3^w) = 3T(3^{w-1}) + \log_3(3^w) = 3T(3^{w-1}) + w$$

abiego resolviendo por pasos

Raíz homogénea

$$t_w - 3t_{w-1} = 0$$

abiego obtenemos la raíz $r_0 = 3$ con multiplicidad $m_0 = 1$

Raíz la función de ajuste, claramente $w = 1^w$ es lo que $r_1 = 1$ con $m_1 = 2$

$$\text{Por tanto } T(u) = c_0 3^u + (c_1 + c_2 u) \Leftrightarrow T(u) = c_0 3^{\log_3 u} + (c_1 + c_2 \log_3 u) = c_0 u + c_1 + c_2 \log_3 u.$$

Es decir, $T(u) \in O(u)$.

2. (2 puntos) Una empresa de gestión de aguas ha encontrado un nuevo pozo capaz de abastecer a una serie de N pueblos de una comarca. Para diseñar la estrategia de abastecimiento deciden calcular el coste de interconectar (mediante tuberías) cada par de pueblos x e y entre sí, $c(x, y)$, y cada pueblo directamente con el pozo p , $c(x, p)$. Diseñad un algoritmo para que dicha empresa pueda abastecer de agua a los N pueblos usando ese pozo de manera que el coste de construir toda la infraestructura sea el mínimo. El algoritmo debe encontrar la solución óptima y además hacerlo de forma eficiente.

Pues nos están pidiendo conseguir el árbol generador minimal usando el algoritmo de Kruskal. En el examen lo explicaría, recordaría por qué da la solución óptima y el sentido de los costos. Además recordaría por qué es más eficiente que Prim, es un gráfico poco denso. Esto no se sabe pero se intuye, en caso de no serlo usaría Prim.

3. (2 puntos) Sea T un conjunto de n programas, cada uno con un tamaño t_1, t_2, \dots, t_n (los números t_i son enteros positivos), y disponemos de un dispositivo de almacenamiento de capacidad máxima C (también un número entero positivo). Se pretende encontrar la manera de decidir qué programas (completos) almacenar en el dispositivo de manera que se maximice el espacio ocupado del dispositivo. Diseñad un algoritmo de programación dinámica para resolver este problema. Aplicadlo para el caso en que $n = 4$, el vector de tamaños es $t(1) = 3, t(2) = 7, t(3) = 11$ y $t(4) = 8$, con capacidad del dispositivo $C = 12$. Algoritmo de la mochila

Este ejercicio es clavado al ejercicio de los teuros y los exámenes luego la implementación basada en PD o la misma así como la destrucción del POB. Simplemente, deberemos

cambiar veces por programas y días por capacidad

5. (2 puntos) Dados n números positivos w_i , $1 \leq i \leq n$, y un entero más M , se trata de encontrar todos los subconjuntos de números w_i cuya suma valga exactamente M . Por ejemplo, si $n = 4$, ($w_1 = 11, w_2 = 13, w_3 = 24, w_4 = 7$) y $M = 31$, entonces los subconjuntos buscados son $(11, 13, 7)$ y $(24, 7)$. Se pide realizar un algoritmo de exploración en grafos para solucionar el problema.

L. Duran. Taller...

Para ello usaremos Backtracking pues para mí es más fácil. Bastará con programar la clase Solución y aplicar el esquema que hay en los apuntes.

Puede ocurrir que no le vea sentido a aplicar una cota inferior más que sumar el menor de lo restante a lo que ya tenemos para ver si superamos la cifra. Si llegamos cumpliendo con una cota local

class Solución {

private:

vector<int> comp;

vector<int> sol;

int u;

public:

Solución (vector<int>&comp, int N) {

if $u > 0$

$N = N$

comp = comp

v.resize(comp.size())

{

int size() {

return v.size(), //El resultado es
v, no obstante,
comp habrá sido

void ProcesaSolución () {

if !EsSolución()

para cada i de 0 hasta $\text{size}() - 1$

si $v[i] == 1$

cout << comp[i] + "+"

{

bool EsSolución () {

int s = 0;

para cada i de 0 hasta $\text{size}()$

si $v[i] == 1 \rightarrow s += comp[i]$

return $s == M$

```

void InicPcup (int u) {
    si Procesa(u)
        v[u]=0;
    {
}

```

```

void SigValPcup (int u) {
    si Procesa(u)
        v[u]++;
    {
}

```

```

bool TodosJuevados (int u) {
    si Procesa(u)
        return v[u]>1;
    {
}

```

```

void Busc(int u) {
    si Procesa(u)
        v[u]=0;
    {
}

```

```

bool Factible () {
    int s=0;
    para cada i de 0 hasta u
        si v[i]==1 → s+=ocup[i]
    return s≤u
}

```

4. (2 puntos) Se desea conocer si, dado un número natural n , éste es o no un número combinatorio del tipo $\binom{k}{2}$, para algún número k . Por ejemplo, $36 = \binom{9}{2}$, o $499500 = \binom{1000}{2}$. Especificad un algoritmo lo más eficiente posible para resolver este problema. Nota: $\binom{k}{2} = \frac{k*(k-1)}{2}$.

$$\hookrightarrow D_y U \quad z_u = u(u-1) \quad \boxed{U = \sqrt{z_u}}$$

Según las notas ya expuestas, queda claro que el problema se reduce a ver si existen dos enteros consecutivos tales que $z_u = u \cdot (u-1)$. Este problema se resuelve por búsqueda binaria y probando. El algoritmo se dejará leer, usar Pivotaje Quicksort es bastante serisible.