Estructuras de Datos 2 - ST0247
Primer Parcial

Nombre .....
Departamento de Informática y Sistemas
Universidad EAFIT

Marzo 22 de 2018

1 Permutaciones 30%

Calcular las permutaciones de un conjunto de enteros es un problema que surge continuamente en la Ingeniería de Sistemas. Calcular tales permutaciones es importante porque muchos problemas se resuelven simplemente probando cada una de las permutaciones para cierto conjunto de datos. Es por esto que a usted se le ha encomendado la tarea de encontrar todas las posibles permutaciones (no necesariamente en orden ***Lexicográfico***) que se pueden generar dado un conjunto de enteros. Al siguiente algoritmo le falta completar algunas líneas, por favor completa cada una de ellas. Un **ejemplo** de salida para e = {1, 2, 3} sería:

- 1, 2, 3
- 1, 3, 2
- 2, 1, 3
- 2, 3, 1
- 3, 1, 2
- 3, 2, 1

Tengan en cuenta que el método `add(e)` agrega un elemento e al final de una lista y el método `removeLast()` elimina el último elemento de una lista.

```
1  private void permutar(int[] e,
2      boolean[] incluido,
3      LinkedList<Integer> lista){
4    if(lista.size() == e.length){
5      imprimirPerm(lista);
6      return;
7    }
8    for(int i = 0; i < e.length; ++i){
9      if(incluido[i] == ........){
10       continue;
11     }
12     incluido[i] = true;
13     lista.add(e[i]);
14     permutar(e);
15     incluido[i] = ..............;
16     lista.removeLast();
17   }
18 }
19
20 public void permutar(int[] e) {
21   permutar(e,
22            new boolean[e.length],
23            new LinkedList<Integer>());
24 }
```

(a) (10%) Complete la línea 13 .....

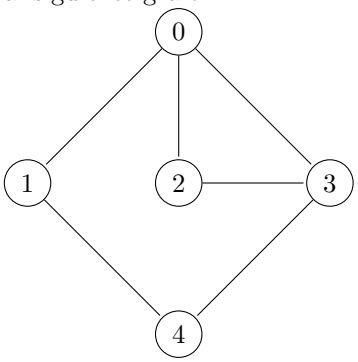
(b) (10%) Complete la línea 19 .....

(c) (10%) Asumiendo que *imprimirPerm*(*lista*) se ejecuta en $O(1)$. ¿Cuál es la complejidad asintótica, en el peor de los casos, del algoritmo anterior?

(i) $O(n^2)$
(ii) $O(2^n \times n)$
(iii) $O(n!)$
(iv) $O(n^4)$

2 DFS & BFS 20%

Los ejercicios de esta sección se deberan resolver de acuerdo al siguiente grafo.



(a) (10%) DFS. Un posible recorrido ***DFS*** del grafo anterior, al ejecutarlo desde el vértice 0, es:

(i) 0, 4, 1, 2, 3
(ii) 0, 2, 4, 3, 1
(iii) 0, 1, 4, 3, 2
(iv) 0, 4, 2, 3, 1

(b) (10%) BFS. Un posible recorrido ***BFS*** del grafo anterior, al ejecutarlo desde el vértice 0, es:

(i) $0, 1, 2, 3, 4$
(ii) $0, 1, 4, 2, 3$
(iii) $0, 4, 3, 2, 1$
(iv) $0, 4, 2, 1, 3$

3 Fuerza bruta 20%

Tenemos dos enteros, M y N. Queremos contar en el rango $[M, N]$ todos los enteros que contienen los dígitos a, b, c, d y e, $1 \leq a, b, c, d, e \leq 9$. Al siguiente algoritmo le faltan algunas líneas. Ayudanos a completarlas.

- **Ejemplo:** Considere $a = 1, b = 1, c = 4, d = 1, e = 4$ y el rango $[200, 300]$. Los enteros que cumplen la condición son: $214, 241$. Por lo tanto la solución debería ser 2.
- **Nota:** Observe que a, b, c, d, e no tienen que ser necesariamente distintos; además sólo importa que a, b, c, d, e estén todos en el numero, es decir, si $S = \{d_1, d_2, d_3, ..., d_k\}$ es el conjunto de todos los dígitos de algún numero $N \leq x \leq M$ y $T = \{a, b, c, d, e\}$, siempre se cumple que $S \cap T = T$.

```
1  public int contar(int N, int M){
2    int total = 0;
3    int[] set;
4    for(int i = N; i <= M; ++i){
5      set = new int[10];
6      int temp = i;
7      while(temp > 0){
8        int rem = ____________;
9        set[rem] = set[rem] + 1;
10       temp = temp / 10;
11     }
12     if(set[a] > 0 && set[b] > 0 &&
13       set[c] > 0 && set[d] > 0 &&
14       set[e] > 0){
15       total = total + 1;
16     }
17   }
18   return total;
19 }
```

(a) (10%) Complete la línea 8 .....

(b) (10%) ¿Cuál es la complejidad del algoritmo?.

(i) $O(|N - M|)$
(ii) $O(|N - M| \times \log_{10} M)$
(iii) $O(|N - M| \times N)$
(iv) $O(|N - M| \times M)$

4 Voraces 30%

Considere el siguiente problema. El doctor le ha recomendado comer a vitaminas, b proteínas y c minerales. Usted puede ir a un lugar donde hay infinitas cantidades de éstas tres cosas; además, usted no tiene que pagar por nada. Sin embargo, uno puede ir sólo una vez por día y cada día que uno va al lugar tiene que tomar una de estas dos posibilidades:

1. Tomar 3 cosas distintas (vitaminas, proteínas y minerales), es decir puede tomar una vitamina, una proteína y un mineral;
2. Tomar 2 cosas del mismo tipo, es decir, 2 vitaminas, 2 proteínas o 2 minerales.

Como usted está muy enfermo quiere saber cuál es la mínima cantidad de días que tiene gastar para ir a ese lugar para tener al menos a vitaminas, b proteínas y c minerales.

Ejemplo: Suponga que $a = 3, b = 4, c = 7$. La solución óptima sería ir los tres primeros días por 3 cosas distintas. Luego los siguientes días ir por 2 cosas iguales. Si se hace el cálculo se demostrará fácilmente que usted tiene que ir 3 días seguidos por dos cosas distintas. Por lo tanto la solución sería 6 días. Es decir, un posible recorrido con el mínimo número de viajes es el siguiente:

1. Ir por 1 vitamina, 1 proteína y 1 mineral
2. Ir por 1 vitamina, 1 proteína y 1 mineral
3. Ir por 1 vitamina, 1 proteína y 1 mineral
4. Ir por 2 proteínas
5. Ir por 2 minerales
6. Ir por 2 minerales

```
1  int numeroDias(int a, int b, int c){
2   //Intentar primero tomando de a 3
3   int minimo = Math.min(a, Math.min(b, c));
4   //Hay que quitarlas a cada variable
5   a = a - minimo;
6   b = b - minimo;
7   c = c - minimo;
8   //Ahora tomemos de a 2 cosas.
9   int temp = a + b + c + 1;
10  temp = ____________________;
11  return ____________________;
12 }
```

- **Pista 1:** La forma óptima de ir por tres cosas cada día es solo cuando tengo que ir por de las tres cosas, es decir, cuando $a > 0 \wedge b > 0 \wedge c > 0$. La mejor forma de ir por dos cosas cada día es cuando solo tengo que ir por a lo sumo dos cosas, es decir, $a = 0 \vee b = 0 \vee c = 0$.
- **Pista 2:** Verifique que $a = minimo \vee b = minimo \vee c = minimo$.

Por favor complete las siguientes líneas:

(a) (10%) Complete la línea 10 .....

(b) (10%) Complete la línea 11 .....

(c) (10%) ¿Cuál es la complejidad del método anterior?

(i) $O(n)$
(ii) $O(1)$
(iii) $O(n^2)$
(iv) $O(n \log n)$