Universidad del Valle

Taller preparatorio segundo examen opcional
FUNDAMENTOS DE ANÁLISIS Y DISEÑO DE ALGORITMOS
Grupo 80 - Duración: 2 horas

Carlos Andres Delgado S, Ing *

11 de Junio de 2014

Nota: Por favor marcar todas las hojas del examen con su nombre y código de estudiante. Recuerde colocar los procedimientos debido a que tienen un gran valor en la calificación.

1. Colas de prioridad, algoritmos de ordenamiento y montones [40 puntos]

1. [**12 puntos**] **HeapSort: Verdadero o Falso** Para cada una de las siguientes afirmaciones decida si es verdadera (V) o falsa (F). Justifique su respuesta

   a) Todo arreglo ordenado decrecientemente es un montón.

   b) Todo montón es un arreglo ordenado decrecientemente.

   c) La complejidad en el peor caso de Heapify sobre un arreglo de tamaño n es $O(lgn)$.

   d) La complejidad del HeapSort sobre un arreglo ordenado decrecientemente es $\Theta(1)$.

2. [**8 puntos**] Un estudiante de una prestigiosa universidad asegura que tiene una implementación de colas de prioridad, que soporta las operaciones de *Inserción*, *Máximo*, *Retirar-Máximo*, todas en $O(1)$ en el peor caso. Argumente por qué esa implementación tiene que tener un error.

3. [**20 puntos**] Sea A un arreglo que representa un montón. Sea B el montón resultante de aplicar la siguiente secuencia de operaciones sobre A:

   - m Maximo $M(A)$
   - Retira Maximo M(A)

   Para las siguientes afirmaciones indique si es verdadero o falso, argumente su respuesta. En caso de ser falso dibuje el montón resultante:

   a) Si A está ordenado descendientemente como arreglo entonces A y B son iguales.

   b) A y B son distintos siempre

   c) Existen montones, no ordenados descendientemente, tales que A es igual a B

   d) A y B son iguales siempre.

*carlos.andres.delgado@correounivalle.edu.co

2. Programación dinámica y programación voraz [60 puntos]

2.1. Conceptos teóricos [20 puntos]

1. [**5 puntos**] ¿Cual es la diferencia entre programación dinámica y el metodo de dividir y conquistar?.

2. [**7 puntos**] Indique los pasos para desarrollar un algoritmo usando programación dinámica.

3. [**8 puntos**] Indique las diferencias entre programación dinámica y programación voraz.

2.2. Análisis de algoritmos [40 puntos]

El algoritmo de Kruskal se utiliza en teoría de grafos para encontrar el árbol recubridor de costo mínimo T de un grafo conexo G. El procedimiento del algoritmo es el siguiente:

1. Se elige un vértice v de G y se considera $T = v$

2. Se considera la arista de peso mínimo e que al agregar a T no genere un ciclo y un vértice que no es de T se hace $T = T \cup e$

3. Si el número de aristas de T es $n-1$ el algoritmo termina. En caso contrario volver al paso 2.

El pseudo-código del algoritmo puede verse a continuación:

```
function Kruskal(G)
  Para cada v en V[G] hacer
       Nuevo conjunto C(v) = {v}.

  //Q contiene todas las aristas
  Nuevo monton Q
  T = vacio
  // n es el numero total de vertices
  Mientras T tenga menos de n-1 vertices
      hacer
    (u,v) = Q.sacarMin()
    // previene ciclos en T. agrega (u,v)
    // si u y v á estn diferentes
    // componentes en el conjunto.
    // Notese que C(u) devuelve
    // la componente a la
    // que pertenece u.
    if C(v) != C(u) then
      Agregar arista (v,u) a  T.
      Merge C(v) y C(u) en el conjunto
Return arbol T
```

1. [**6 puntos**] Aplique el algoritmo de Kruskal para hallar el árbol recubridor de costo mínimo al siguiente grafo G:

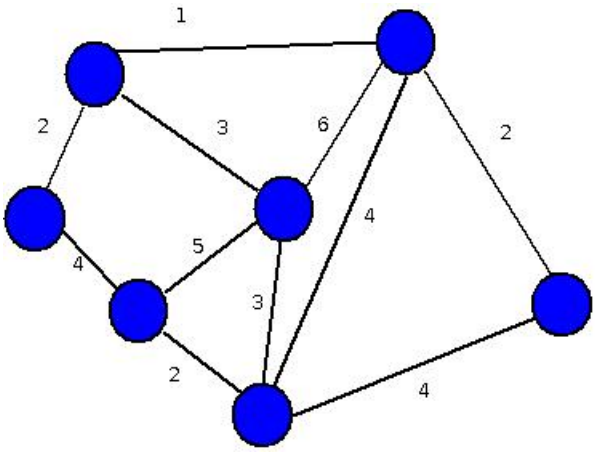

Figura 1: Grafo de ejemplo

2. [**14 puntos**] ¿Este algoritmo siempre da la solución óptima. Justifique su respuesta?.

3. [**20 puntos**] Identifique la estrategia voraz del algoritmo. Explique con un ejemplo.