

1 tercer corte examen

1.1 OrdenamientoI

1. ordenamiento1

En el peor caso del QuickSort ¿Que sucede con el pivote?

- (a) Es el mínimo o máximo de los elementos que deseamos ordenar, por lo que la división del algoritmo es desventajosa (100%)
- (b) Es aproximadamente promedio de los elementos que deseamos ordenar, por lo que la división del algoritmo es desventajosa
- (c) Es la moda de los elementos que deseamos ordenar, por lo que la división del algoritmo es desventajosa
- (d) Es un elemento cualquiera de los elementos que deseamos ordenar, por lo que la división del algoritmo es desventajosa

2. ordenamiento2

En el mejor caso del QuickSort ¿Que sucede con el pivote?

- (a) Es el mínimo o máximo de los elementos que deseamos ordenar, por lo que la división del algoritmo es ventajosa
- (b) Es un valor que genera dos divisiones de aproximadamente $\frac{n}{2}$ en cada llamado. (100%)
- (c) Es la moda de los elementos que deseamos ordenar, por lo que la división del algoritmo es ventajosa
- (d) Es un elemento cualquiera de los elementos que deseamos ordenar, por lo que la división del algoritmo es ventajosa

3. ordenamiento3

¿Indique cuales de los siguientes arreglos son Monticulos?

- (a) $\{10, 8, 9, 5, 3, 4, 7\}$ (33.33333%)
- (b) $\{12, 8, 10, 5, 3, 4, 7\}$ (33.33333%)
- (c) $\{12, 8, 10, 6, 3, 4, 7\}$ (33.33333%)

- (d) $\{10, 3, 9, 5, 3, 4, 7\}$ (-33.33333%)
- (e) $\{12, 8, 10, 11, 3, 4, 7\}$ (-33.33333%)
- (f) $\{12, 8, 10, 11, 9, 4, 7\}$ (-33.33333%)

1.2 OrdenamientoII

1. OrdenamientoII1

Deseo ordenar en tiempo lineal el arreglo $\{1, 40, 400, 1000, 2, 500, 17, 200\}$
¿Cual es el algoritmo que me permite solucionar este problema?

- (a) Counting Sort
- (b) Radix-Sort
- (c) Bucket Sort, Normalizando. (100%)

2. OrdenamientoII2

Deseo ordenar en tiempo lineal el arreglo $\{1, 2, 3, 4, 6, 8, 10, 1, 2, 3\}$. ¿Cual es el algoritmo que me permite solucionar este problema?

- (a) Counting Sort (100%)
- (b) Radix-Sort
- (c) Bucket Sort

3. OrdenamientoII3

Deseo ordenar en tiempo lineal el arreglo $\{123, 223, 312, 412, 623, 832, 102, 132, 234, 323\}$
. ¿Cual es el algoritmo que me permite solucionar este problema?

- (a) Counting Sort
- (b) Radix-Sort (100%)
- (c) Bucket Sort

1.3 ProgDinamicaI

1. ProgDinamicaI1

Marque las afirmaciones que son ciertas con respecto a programación dinámica:

- (a) Se garantiza la solución optima (33.33333%)
- (b) Se tiene una subestructura óptima que almacena respuesta a subproblemas (33.33333%)
- (c) Se utiliza en problemas de optimización (33.33333%)
- (d) No se garantiza solución optima (−33.33333%)
- (e) Se calcula la solución a todos los subproblemas (−33.33333%)
- (f) Se puede utilizar en cualquier problema de computación (−33.33333%)

2. ProgDinamicaI2

☐ MULTI ☐ 1.0 point ☐ 0.10 penalty ☐ Multiple ☐ Shuffle

Se puede decir del problema de la Mochila:

- (a) En cada paso se toma una decisión de llevar o no llevar el elemento (33.33333%)
- (b) Siempre se tienen dos caminos (decisión) y se escoge el que mayor ganancia dé (33.33333%)
- (c) El caso trivial es cuando tenemos cero elementos (33.33333%)
- (d) Siempre decidimos llevar el elemento (−33.33333%)
- (e) El caso trivial es cuando tenemos todos los elementos y decidimos si los llevamos o no (−33.33333%)
- (f) Únicamente tenemos un camino en el cual tenemos la mejor ganancia (−33.33333%)

3. ProgDinamicaI3

☐ MULTI ☐ 1.0 point ☐ 0.10 penalty ☐ Multiple ☐ Shuffle

Se puede decir del problema de la subsecuencia común más larga:

- (a) Si el último carácter de ambas secuencia es igual, el siguiente subproblema es con ambas secuencias sin el último elemento. (33.33333%)
- (b) Si el último carácter de ambas secuencia es diferente, se toma la cadena más grade de dos subproblemas, en los cuales se le quita el último elemento a sólo una de las dos cadenas. (33.33333%)
- (c) El caso trivial es cuando intentamos obtenemos la subsecuencia común más larga entre una cadena vacía y otra cadena. (33.33333%)
- (d) Si el último carácter de ambas secuencia es igual, se toma la cadena más grade de dos subproblemas, en los cuales se le quita el último elemento a sólo una de las dos cadenas. (−33.33333%)

- (e) Si el último carácter de ambas secuencia es diferente, el siguiente subproblema es con ambas secuencias sin el último elemento. (−33.33333%)
- (f) El caso trivial es cuando intentamos obtenemos la subsecuencia común más larga cuando ambas cadenas no son vacías. (−33.33333%)

1.4 ProgDinamicaII

1. ProgDinamicaII1

☐ MULTI ☐ 1.0 point ☐ 0.10 penalty ☐ Single ☐ Shuffle

Dado el problema de la Mochila con $M = 10$, $w = \{3, 4, 2, 5\}$, $b = \{2, 3, 1, 2\}$ ¿Como son las expansiones de los primeros dos llamados?.

- (a) $g_4(10) = \max(\max(g_2(3)+1, g_2(5))+2, \max(g_2(8)+1, g_2(10)))$ (100%)
- (b) $g_4(10) = \max(\max(g_2(3), g_2(5) + 1) + 2, \max(g_2(8), g_2(10) + 1))$
- (c) $g_4(10) = \max(\max(g_2(5), g_2(5) + 1) + 2 + 2, \max(g_2(8), g_2(8) + 1))$
- (d) $g_4(10) = \max(\max(g_2(3), g_2(5) + 1), \max(g_2(8), g_2(10) + 1) + 2)$
- (e) $g_4(10) = \max(\max(g_2(5), g_2(5) + 1), \max(g_2(8), g_2(8) + 1) + 2)$

2. ProgDinamicaII2

☐ MULTI ☐ 1.0 point ☐ 0.10 penalty ☐ Single ☐ Shuffle

Dado el problema de la Mochila con $M = 10$, $w = \{5, 4, 2, 8\}$, $b = \{2, 3, 1, 3\}$ ¿Como son las expansiones de los primeros dos llamados?.

- (a) $g_4(10) = \max(g_2(2) + 3, \max(g_2(8) + 1, g_2(10)))$ (100%)
- (b) $g_4(10) = \max(\max(g_2(2), g_2(2) + 3), \max(g_2(8) + 1, g_2(10)))$
- (c) $g_4(10) = \max(g_2(2) + 3, \max(g_2(8), g_2(10) + 1))$
- (d) $g_4(10) = \max(\max(g_2(2), g_2(2) + 3), \max(g_2(8), g_2(10) + 1))$
- (e) $g_4(10) = \max(g_2(8) + 3, \max(g_2(2) + 1, g_2(8)))$

3. ProgDinamicaII3

☐ MULTI ☐ 1.0 point ☐ 0.10 penalty ☐ Single ☐ Shuffle

Dado el problema de la subsecuencia más larga (LCS) con $A = abcde$ y $B = abbde$ ¿Como son las expansiones de los primeros dos llamados?.

- (a) $LCS(abcde, abbde) = (LCS(abc, abb) + 1) + 1$ (100%)
- (b) $LCS(abcde, abbde) = \max(LCS(abcd, abbde), LCS(abcde, abbd))$
- (c) $LCS(abcde, abbde) = \max(LCS(abc, abbde)+1, LCS(abcd, abbd)+1)$

- (d) $LCS(abcde, abbde) = (LCS(max(LCS(abc, abbd)+1, LCS(abb, abb)+1) + 1)$
 (e) $LCS(abcde, abbde) = max(LCS(abcd, abb)+1, LCS(abcd, abbe))+1$

4. ProgDinamicaII4

Dado el problema de la subsecuencia más larga (LCS) con $A = abcdf$ y $B = abbde$ ¿Como son las expansiones de los primeros dos llamados?.

- (a) $LCS(abcdf, abbde) = max(max(LCS(abc, abbde), LCS(abcd, abbd)), max(LCS(abcd, abbde), LCS(abcd, abb)))$
 (b) $LCS(abcdf, abbde) = max(LCS(abc, abbde), max(LCS(abcd, abbd), LCS(abcdf, abb)))$
 (c) $LCS(abcdf, abbde) = max(max(LCS(abc, abbd), LCS(abcd, abbd)), max(LCS(abc, abbde), LCS(abcd, abb)))$
 (d) $LCS(abcdf, abbde) = max(LCS(abc, abbde)+1, max(LCS(abcd, abbd), LCS(abcdf, abb)))$
 (e) $LCS(abcdf, abbde) = max(LCS(abc, abbde)+1, LCS(abcd, abbd)+1)$

1.5 ProgVoraz

1. ProgVoraz1

Marque las afirmaciones que son ciertas con respecto a la programación voraz

- (a) No se garantiza solución optima (33.33333%)
 (b) Sólo de trabaja con la mejor solución de los problemas locales sin tomar en cuenta el global (33.33333%)
 (c) Aplica para problemas que se pueden solucionar con programación dinámica (33.33333%)
 (d) No se garantiza solución optima (−33.33333%)
 (e) Se trabaja considerando la solución global del problema en cada paso (−33.33333%)
 (f) Aplica para cualquier problema que se puede solucionar en un computador (−33.33333%)

2. ProgVoraz2

Marque las afirmaciones que son ciertas con respecto a la solución voraz de la programación de actividades (vista en clase).

- (a) No se garantiza solución optima (33.33333%)
- (b) Se ordenan las tareas de acuerdo a su tiempo de finalización, para así asignar el recurso (33.33333%)
- (c) Se busca colocar primero las tareas que **terminan primero** para dar espacio a las siguientes tareas (33.33333%)
- (d) Se garantiza solución optima (−33.33333%)
- (e) Se ordenan las tareas de acuerdo a su tiempo de inicio, para así asignar el recurso (−33.33333%)
- (f) Se busca colocar primero las tareas que **inician primero** para dar espacio a las siguientes tareas (−33.33333%)

3. ProgVoraz3

MULTI

1.0 point

0.10 penalty

Multiple

Shuffle

Marque las afirmaciones que son ciertas con respecto a la solución voraz de la mochila (vista en clase).

- (a) No se garantiza solución optima (33.33333%)
- (b) Se colocan primero las tareas cuya relación entre beneficio y peso sea la mejor (33.33333%)
- (c) El costo de la solución es $O(n \log(n))$ por el ordenamiento que se debe realizar (33.33333%)
- (d) Se garantiza solución optima (−33.33333%)
- (e) Se colocan primero las tareas que ofrezcan mayor beneficio, sin importar su peso (−33.33333%)
- (f) El costo de la solución es $O(n)$ porque solo se necesita recorrer la estructura que tiene los elementos (−33.33333%)

Total of marks: 16