Metodoloxía e Tecnoloxía da Programación
1º ETIS. Curso 2007/08
Cuestionario xeral 2º cuadrimestre

Escola Técnica Superior de Enxeñería

As cuestións deste cuestionario son un repaso xeral aos temas mais relevantes do segundo cuadrimestre. Moitas delas son preguntas con opcións de varias respostas (tipo test)Nalgúns casos non son preguntas tipo test, senón que teñen unha resposta un pouco mais ampla. Polo tanto non serán exactamente do estilo
Nalgúns casos son cuestións moito mais amplas das que se van facer no exame.

TEMA 8. RECURSIÓN

1. Consideremos o procedemento recursivo

```
void triangulo(char n){

     if(n<0)
      return;
     else{
          triangulo(n-1);
          printf("%d", n);
          }
     }
```

- Indicar cal é a súa saída cando se invoca coa chamada **triangulo(5)**

01235

- ¿De que tipo de recursión se trata?

Recursión lineal no final

- De ser posible, convertilo a un deseño iterativo equivalente

Al ser un tipo de recursión lineal no final no se ajusta al esquema de traducción planteado en la transparencia 8.3.40. Por tanto se debería implementar simulando mediante una pila la pila de recursión del sistema.

2. Considerando función recursiva que se indica de seguido, que calcula os elementos do **triángulo de Pascal**, definido para k≤n como:

$$tp(n,k) = \begin{cases} 1 & k = 0 \text{ ó } n = k \\ tp(n-1,k-1) + tp(n-1,k) & otro\ caso \end{cases}$$

- Construir a árbore de activacións para **tp(4, 2)**, indicando a orde de obtención dos termos

O triángulo de Pascal para o caso (4, 2) é:

	k=0	k=1	k=2
n=0	1		
n=1	1	1	
n=2	1	2	1
n=3	1	3	3
n=4	1	4	6

- **Subliñamos os casos non recursivos**

4,2
3,1 **3,2**
2,0 **2,1** **2,1** **2,2**
1,0 **1,1** **1,0** **1,1**

A orde de obtención dos termos será recorrendo en profundidade a árbore:
2,0 1,0 1,1 2,1 3,1 1,0 1,1 2,1 2,2 3,2 4,2

- Aplicar unha técnica de programación dinámica ao deseño e estimar o consumo de memoria adicional que supón.

Á vista do anterior, hai termos que se obteñen varias veces, polo que é apropiado utilizar unha técnica dinámica. Unha posible implementación C sería:
Cunha táboa global unsigned char tabla[N][N], con N máximo n que é posible tratar:

```
unsigned char tp (unsigned char n, unsigned char k){
if(tabla[n][k]==0)
        if (k==0 || k==n)
          tabla[n][k]=1;
        else if(k!=0 && k!=n)
          tabla[n][k]=tp(n-1, k-1)+tp(n-1, k);
        else{
          printf("ERROR");
          exit(0);
        }
return tabla[n][k];
```

O tamaño da táboa sería neste caso N^2, xa que o tamaño das filas ten que ser constante.

- **Se optamos por unha implementación dinámica podemos facer mais eficiente o tamaño da táboa, xa que non é preciso que cada fila teña tamaño N. De feito, se nos fixamos no seguinte exemplo:**

	k=0	k=1	k=2	k=3	k=4
n=0	1				
n=1	1	1			
n=2	1	2	1		
n=3	1	3	3	1	
n=4	1	4	6	4	1

Triángulo de Pascal para n=4

- **o tamaño de cada fila é k+1. Polo tanto, o caso peor será n=k. Como o número de filas é sempre n+1, temos que o tamaño máximo da táboa será para o caso peor:**

$$T = \sum_{i=0}^{n} (i+1) = \sum_{i=0}^{n} i + \sum_{i=0}^{n} 1 = \frac{n\cdot(n+1)}{2} + (n+1) = \frac{n(n+3)}{2} + 1$$

3. Indica cal é a secuencia de activacións para a solución seguinte ao problema do cambio de moedas:

$$numMonedas(i, importe) = \begin{cases} 0 & importe = 0 \\ \text{sin } solución & i = 0, importe < euros[0] \\ 1 + numMonedas(i, importe - euros[0]) & i = 0, importe \geq euros[0] \\ numMonedas(i-1, importe) & i > 0, importe < euros[i] \\ \min(numMonedas(i-1, importe), 1 + numMonedas(i, importe - euros[i])) & en\ otro\ caso \end{cases}$$

para o caso do sistema monetario dos euros, cando na máquina quedan únicamente moedas de 1, 5, 20 e 50 céntimos, e ten que devolverlle ao cliente 25 céntimos.

3,25->2,25
2,25->1,25 | 2,5
1,25->0,25 | 1,20
0,25->0,24->0,23->...->0,1->0,0
1,20->0,20|1,15
0,20->0,19->0,18->...->0,1->0,0
1,15->0,15 | 1,10
0,15->0,14->0,13->...->0,1->0,0
1,10->0,10 | 1,5
0,10->0,9->0,8->...->0,1->0,0
1,5->0,5 | 1,0
0,5->0,4->0,3->...>0,1->0,0
2,5->1,5 | 1,0
1,5->0,5 | 1,0
0,5->0,4->0,3->...>0,1->0,0

TEMA 9. ALGORITMIA

1. Indicar para o caso n=17, b=2 cal destas é a saída correcta por pantalla da función:

```
funcion(int n, int b){
 if (n>=b)
  funcion(n/b, b);

 printf("%d ", n%b);
}
```

- 17 15 13 11 ... 1
- **1 0 0 0 1**
- 17 8 4 2 1 0
- 16 1

e cal destas expresións é correcta, referida ao seu tempo de execución t(n):

- $t(n) \in \Theta(n)$
- $t(n) \in \Theta(n^2)$
- $t(n) \in \Theta(\log(n))$

El tiempo de ejecución viene dado por la recurrencia siguiente:

$$t(n,b) = \begin{cases} a_1 + t(n/2,b) & n \geq b \\ a_2 & n < b \end{cases}$$

con: a_1 [constante para n, b pequeños]: tiempo de evaluar n>=b, calcular n/b, calcular n%b y hacer el printf
a_2 [constante para n, b pequeños]: tiempo de evaluar n>=b, calcular n%b y hacer el printf
Resolviendo la recurrencia, tenemos:

$t(n,b) = a_2 + a_1 \log_b n$ **si n es potencia de b. Por tanto,** $t(n,b) \in \theta(\log n / n \; es \; potencia \; de \; b)$ **. Como log n es suave y t(n) es asintóticamente no decreciente, podemos aplicar la regla de la uniformidad, y por tanto** $t(n,b) \in \theta(\log n)$

Algoritmos voraces

1. O sistema de xestión da cola da fotocopiadora da ETSE utiliza unha **estratexia voraz** para despachar os traballos dos usuarios. A dita estratexia consiste en minimizar o tempo de agarda global de todos os traballos. Sabemos que a impresora opera a una velocidad de V (en kB/s) e que nun momendo dato hai agardando K traballos, cada un deles de tamaño (en kB) T_k, k=1,…,K. Un estudante de 1º de ETIS vai imprimir un documento PDF de tamaño T (kB). Revisando a cola da impresora decátase que $T>T_{0.9K}$, así que o tempo de agarda vai ser.

$$TIEMPO_T = \frac{T}{V} + \frac{1}{V}\sum_{i=1}^{0.9K} T_i \; .$$

Indícalle ao estudante algunha estratexia que lle axude a reducir o tempo de agarda.

a) Vamos a trabajar con el mínimo tiempo que tendría que esperar el nuevo usuario. Este será si su trabajo se coloca justo a continuación del trabajo 0.9K. En ese caso, el tiempo de espera será la suma de:

1) El tiempo que tarden todos los trabajos anteriores, que serán los de índice i=1, …, 0.9K

Por tanto, TIEMPO$_1$= $\sum_{i=1}^{0.9K} \frac{T_i}{V} = \frac{1}{V}\sum_{i=1}^{0.9K} T_i$

2) Mas el tiempo que tarde en imprimirse el propio trabajo: T

Es decir, el usuario esperará un tiempo total $TIEMPO_T = \frac{T}{V} + \frac{1}{V}\sum_{i=1}^{0.9K} T_i$ **.**

b) Para reducir ese tiempo de espera, hay que considerar que la estrategia voraz hace lo siguiente: ordena los trabajos por orden creciente de tamaño, despachando el mas pequeño en primer lugar, el siguiente de segundo, …, y el mas grande en último lugar.

Por tanto, el usuario debería intentar que su documento se pusiese delante de alguno de los i=1, …, 0.9K anteriores, siendo lo óptimo que se pusiese de primero (delante de todos). Una forma de hacerlo sería dividir su documento en N fragmentos (obviamente que sean de un número entero de páginas), para conseguir que alguno de los fragmentos se ponga delante de algunos de los anteriores. Idealmente si escogemos un N suficientemente grande, cada fragmento tendría un tamaño T/N<T$_1$ y así todos sus fragmentos de trabajo saldrían en primer lugar.

NOTA: Habría que valorar adicionalmente el tiempo que le llevase fragmentar el trabajo y luego ordenar las hojas cuando las saque de la impresora, pero ambas tareas son casi inmediatas en este problema: en la opción de imprimir su documento va escogiendo intervalos de páginas y lo hace ordenadamente (desde la primera hasta la última), las hojas saldrán ordenadas en la impresora y no consumirá tiempo en reordenar los fragmentos. Evidentemente, si N es muy grande, la tarea de fragmentar el trabajo sería costosa (además de aburrida y sujeta a errores –páginas repetidas, …-), de modo que sería bueno encontrar un N intermedio que adelantase bastante la salida del trabajo en la impresora sin que haga muy lenta la fragmentación del documento.

2. Indicar cal destas afirmacións é certa:

   - O problema do cambio de moedas no sistema monetario dos euros sempre ten solución voraz
   - **O problema do cambio de moedas no sistema monetario dos euros sempre ten solución recursiva óptima**
   - O problema da mochila (un obxecto de cada clase) sen fraccionar obxectos sempre ten solución voraz óptima
   - **O problema da mochila (un obxecto de cada clase) fraccionando obxectos sempre ten solución voraz óptima**
   - **O problema da mochila (un obxecto de cada clase) fraccionando obxectos ten solución recursiva óptima**

Metodoloxía e Tecnoloxía da Programación
1º ETIS. Curso 2007/08
Cuestionario xeral 2º cuadrimestre

Escola Técnica Superior de Enxeñería

Programación dinámica

1. Considerando o seguinte deseño dinámico para o problema de Fibonacci (transparencia 9.5.16):

```
unsigned long fibonacciRecursivoDinamico(int n, unsigned long *pvalores){
//pvalores apunta al valor que se quiere calcular
if (*pvalores==0)
 if(n==0 || n==1)
   *pvalores=(unsigned long)1;
 else
  *pvalores=fibonacciRecursivoDinamico(n-1, pvalores-1)+
            fibonacciRecursivoDinamico(n-2, pvalores-2);

return *pvalores;
}
```

pídese facer unha estimación do custe en operacións e memoria para o caso peor e mellor.

- **Número de operacións:**
- **Caso mellor: 1**
- **Caso peor: elemento n non calculado e ningún dos anteriores tampouco. Neste caso dase a execución recursiva lineal múltiple convencional, xa analizada en teoría. Orden exponencial**

$$t(n) \in \Theta\left(\left(\frac{1+\sqrt{5}}{2}\right)^n\right)$$

- **Memoria:**
- **Caso mellor: xa calculado, polo que a táboa non cambia**
- **Caso peor: precisa unha táboa de n elementos enteiros.**

$$t(n) \in \Theta(n)$$

2. Indicar, no caso de optar por unha implementación dinámica da función 8.3, o seguinte:
   - como se reduce a secuencia de activacións

Subliñamos as activacións que xa non se realizan:
3,25->2,25
2,25->1,25 | 2,5
1,25->0,25 | 1,20
0,25->0,24->0,23->...->0,1->0,0
1,20->0,20|1,15
- **0,20->0,19->0,18->...->0,1->0,0**

1,15->0,15 | 1,10
- **0,15->0,14->0,13->...->0,1->0,0**

1,10->0,10 | 1,5
- **0,10->0,9->0,8->...->0,1->0,0**

1,5->0,5 | 1,0
- **0,5->0,4->0,3->...>0,1->0,0**

2,5->1,5 | 1,0
- **1,5->0,5 | 1,0**
- **0,5->0,4->0,3->...>0,1->0,0**
  - cal sería o contido da táboa ao remate da execución

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
1						1					2					3					4					5
2						1																				2
3																										2

Programación con volta atrás

1. Considerando o esquema de programación con volta atrás aplicado ao problema do comercial da empresa ETISSA, tal e como se describe no guión de prácticas 8.3, indicar se as seguintes afirmacións son verdadeiras ou falsas:
   - o primeiro recorrido analizado é: [0, 1, 2, 3, 4, 5, 6] □ **V** **X F**
   - o quinto recorrido analizado é: [5, 0, 1, 2, 6, 4, 3] □ **V** **X F**
   - o último recorrido analizado é: [5, 6, 5, 4, 3, 2, 1] □ **V** **X F**
   - a solución óptima é única: □ **V** **X F**
   - non pode haber mais de dúas solucións óptimas: □ **V** **X F**

2. O comercial de ETISSA acaba de sair de Santiago e dispón dunha pequena cantidade de combustible, que só lle permitirá recorrer **d** km. O seu navegador GPS incorpora unha funcionalidade (pouco optimizada) para indicarlle a ruta que debería seguir sen quedar tirado na estrada sen combustible. Para iso lle da como resposta o **primeiro percorrido para o que a distancia sexa menor que d, baseándose no** esquema 2 de programación con volta atrás. ¿Como se modificaría a implementación para incorporar a dita funcionalidade?

El paso del esquema 1 al esquema 2 únicamente requiere añadir una bandera que indique cuando se ha encontrado el recorrido que cumple la condición requerida.

Por tanto, cada vez que se encuentre un recorrido completo será necesario evaluar la distancia recorrida, y cuando encontremos una que sea menor que d, modificaremos la bandera para que no se sigan realizando mas búsquedas. Será necesario asumir que disponemos de una función "distancia" que nos devuelve la distancia recorrida por un recorrido completo.

También es necesario modificar el prototipo de la función para pasar por referencia la bandera, que habrá sido inicializada a 0.

Por tanto, la cabecera de la función pasará a ser:

```
void backtracking (unsigned int distancias[NUMERO_CIUDADES][NUMERO_CIUDADES], int visitada[], int k, int recorrido[], char *finalizado) //finalizado es la bandera, por referencia
```

y el if donde se evalúa la respuesta pasará a ser:

```
if (k==NUMERO_CIUDADES-1 && !visitada[i] && *finalizado==0){
 //Sexta ciudad, no visitada
    if(distancia (recorrido) <d) { //ya la tengo
          *finalizado=1;      //Cambio la bandera
```

TEMA 10. ORDEACIÓN E BUSCA

1. Discute, na implementación realizada para o algoritmo do comercial de ETISSA, as vantaxes e inconvenientes que ten realizar a busca da nova cidade a visitar mediante a tabla dinámica visitada fronte a facelo por:
   - Busca lineal
   - Busca dicotómica

Coa táboa dinámica, cada busca se realiza en tempo constante: $t(n)\in\Theta(1)$ cun custe de memoria lineal no número de cidades: $m(n)\in\Theta(nc)$

A busca lineal invirte os termos, tendo $t(n)\in\Theta(nc), m(n)\in\Theta(1)$

A busca dicotómica esixe ter ordeadas as cidades para facer unha busca, polo que $t(n)\in\Theta(\log(nc)), m(n)\in\Theta(1)$. Habería que ter en conta o custe de manter a ordeación no conxunto de cidades non visitadas (o que esixe facer inserciónss e eliminacións nel, cun custe lineal alomenos).

Tendo en conta que no algoritmo é prioritario o tempo de execución a opción dinámica é a mellor sempre que os requisitos do número de cidades non superen os recursos de memoria. Mesmo nese caso sería ainda mais decisiva a táboa de distancias, que ten un tamaño $m(n)\in\Theta(nc^2/2)$

2. A consultora PAQUETIS está a deseñar un sistema de consultas para a base de documentos da biblioteca da ETSE, na que os usuarios poden facer buscas por todos os criterios dos datos de cada documento (título, nome e apelidos do autor, ...) excepto o contido do texto. A consultora propón un deseño baseado na busca dicotómica, xa que o consultor ten información que é a mais eficiente. Discutir esa proposta en termos de eficiencia para ver se é axeitada nos seguintes tres escearios:
   - as actualizacións da base de documentos realízanse só cada quince días
   - as actualizacións de documentos realízanse en calquera momento, coa chegada de novas adquisicións de revistas ou libros
   - durante o mes de agosto e os días festivos, coas vacacións do persoal, non hai actualizacións na base de documentos

   Considerar para todos os escearios que hai un promedio de N buscas diarias na base de documentos por parte dos usuarios.

A busca dicotómica esixe ter ordeada en todo momento a base de n documentos, polo que será unha boa opción cando entre ordeación e ordeación cada unha cun custe $t(n)\in\Theta(nlog(n))$ se realicen un número de consultas suficiente, xa que estas serían en tempo $t(n)\in\Theta(\log(n))$ e non en tempo $t(n)\in\Theta(n)$. Nunha análise xeral, teriamos que:

- **Esceario 1 (actualizacións cada 15 días). O número de buscas entre ordeación e ordeación será 15N e cada unha se faría en $t(n)\in\Theta(\log(n))$. Polo tanto, se $15N\log(n)<nlog(n)$ a opción proposta pola consultora sería apropiada. Iso obriga a que $N<n/15$.**

- **Esceario 2. Esixe unha reordeación con cada nova adqusición. SE estas son frecuentes, non se chegará ao mínimo número de buscas entre reordenacións, polo que a proposta da consultora non sería boa.**
- **Esceario 3. Neste caso ao non haber actualizacións, non é preciso reordenar. Se nun deses períodos é previsible que o número de buscas sexa suficientemente numeroso a proposta da consultora sería boa. Terían que darse as condicións seguintes:**
- **fin de semana (2 días):** $N < n/2$
- **agosto (31 días):** $N < n/31$
- **un día festivo (1 día):** $N < n$