CC40A - Control 1

24 de Mayo de 2006

P1 (1.8 pt)

Motivación: Para buscar una cadena $P = p_1 \dots p_m$ en un texto dado permitiendo k errores decidimos comenzar por partirlo en $k+1$ subpatrones P_i, $P = P_1 \cdot P_2 \cdots P_{k+1}$, y buscar los subpatrones en forma exacta usando algún índice, para luego verificar la vecindad en el texto de cada ocurrencia exacta. El costo del procedimiento final está dado esencialmente por la cantidad de ocurrencias a verificar.

Problema: Por ello, se desea particionar P en $k+1$ subpatrones de manera de minimizar el total de ocurrencias a verificar. Para este fin se tiene en una matriz $T[i, j]$ la cantidad de ocurrencias del substring $p_i \dots p_j$ de P en el texto. Diseñe un algoritmo de programación dinámica para determinar la mejor partición (es decir, que minimice la suma de ocurrencias de los $k + 1$ subpatrones) y analícelo.

Ejemplo: Para buscar `hola` con $k = 1$ las particiones posibles de P son (`h`,`ola`), (`ho`,`la`), o (`hol`,`a`).

P2 (1.8 pt)

El algoritmo de Knuth, Morris y Pratt (KMP) para búsqueda en texto funciona mediante mantener dos punteros, uno al texto y uno al patrón. Según el resultado de las comparaciones, se mueven los punteros `i` (al texto) y `j` (al patrón). Texto (`text`) y patrón (`pat`) terminan con un carácter 0.

```
i = 0; j = 0;
while (text[i] != 0)
   { if (j == -1 || pat[j] == text[i])
        { i++; j++;
          if (pat[j] == 0) report i-j;
        }
     else j = next[j];
   }
```

La tabla `next` satisface $-1 \leq$ `next[j]` $<$ `j` y no importa aquí cómo se calcula.

Note que el algoritmo avanza `i` por todo el texto, y que `i` sólo crece. Sin embargo, puede permanecer cierto tiempo sin avanzar mientras se ejecuta `j = next[j]` repetidamente (con lo cual `j` va decreciendo). Siendo n el largo del texto y m el largo del patrón, un primer análisis indica que el tiempo es $O(mn)$.

Utilice una técnica de análisis amortizado para demostrar que el tiempo es $O(n)$.

P3 (1.8 pt)

Usted va a vivir por una temporada a una ciudad extranjera, donde el boleto de metro cuesta \$1. Existe un carnet de descuento que cuesta $\$D$, y con el cual el boleto de metro se adquiere a $\$B < 1$. Usted realizará una cantidad n de viajes en metro durante su estadía, pero no tiene idea de qué día volverá y por lo tanto no tiene ninguna estimación confiable de n.

Usted podría comprar el carnet el primer día y pagaría $D+nB$, o no comprarlo nunca y pagaría n, o en general comprarlo luego de i viajes pagando en total $D + i + B(n - i)$.

Diseñe una estrategia online y analice su competitividad con respecto al algoritmo óptimo (que sabe cuántos viajes realizará). Demuestre que no se puede obtener mejor competitividad que la de su método.

P4 (1.8 pt)

Se tiene una secuencia de bits y se desea realizar la operación *rank* eficientemente, pero también queremos permitir la operación de invertir un bit de la secuencia.

Diseñe una estructura de datos que permita realizar $rank(i)$ e $invertir(i)$ (que invierte el i-ésimo bit de la secuencia) en tiempo $O(\log n)$, utilizando $O(n)$ bits de espacio.

Hint: comience con un árbol balanceado que contenga un nodo por bit, luego reduzca el espacio.

Sí, el control suma 8.2 puntos. Se eliminará lo que sobre de 7.0.

Tiempo: 2 horas

Una hoja manuscrita de apuntes