Práctica Planificación de Tareas

Objetivos:

- Practicar la estrategia algorítmica voraz para un problema real.

Programa a realizar:

Un procesador tiene que atender a n procesos pero en un determinado momento sólo puede estar ejecutando uno de ellos. Se conoce de antemano el tiempo que necesita cada proceso. Los identificadores de procesos y tiempos asociados están disponibles en un fichero de entrada que deberá leer el programa principal. El nombre del fichero de entrada se proporcionará al programa como argumento en la línea de comandos.

Este fichero tiene la siguiente estructura (se adjunta un ejemplo para 3 procesos pero el programa debe ser general y poder trabajar con cualquier número de procesos):

Ejemplo de fichero de entrada:

P1 5

P2 10

P3 3

En el ejemplo P1 ocuparía 5 unidades de tiempo, P2 10 unidades de tiempo y P3 3 unidades de tiempo. Se quiere determinar en qué orden ha de atender el procesador a los procesos para que se **minimice la suma del tiempo que los procesos esperan a ejecutarse**.

Se trata de un problema que no está sometido a restricciones y una solución es una secuencia de tamaño n en la que aparecen los n procesos a atender. Una estrategia de fuerza bruta obtendría la permutación de las n tareas que minimice la función objetivo. El orden de aparición de una tarea en la permutación indicará el orden de atención por parte del procesador.

Para el ejemplo anterior, se muestra a continuación la suma del tiempo que los procesos están esperando en el sistema para cada una de los seis posibles órdenes en que se podrían atender las 3 tareas:

Orden de Atención	Tiempo de Espera
P1 P2 P3	5 + (5 + 10) = 20
P1 P3 P2	5 + (5 + 3) = 13
P2 P1 P3	10 + (10 + 5) = 25
P2 P3 P1	10 + (10 + 3) = 23
P3 P1 P2	3 + (3 + 5) = 11

Práctica Planificación de Tareas

P3 P2 P1	3 + (3 + 10) = 16

En este ejemplo, la ordenación que produce el tiempo de espera mínimo es la P3 P1 P2.

El alumno deberá desarrollar un programa que lea un fichero de entrada con cualquier número de tareas y planifique tareas de dos modos: a) **estrategia voraz seleccionando tareas en orden creciente de tiempo de proceso** y b) **estrategia voraz seleccionando tareas en orden decreciente de tiempo de proceso**. Cada uno de los modos de funcionamiento será una opción en el programa principal y resultado de la ejecución del mismo el programa mostrará la secuencia elegida así como su tiempo de espera

Pasos a seguir:

- Se sugiere almacenar los datos del fichero de entrada en un vector dinámico cuyo tipo de datos para sus elementos es un struct que guarde los dos datos de cada proceso (cadena de caracteres que identifica el proceso y tiempo del proceso). Será necesario mirar primero en el fichero cuantos procesos hay y luego generar dinámicamente memoria para tantos datos como procesos hay en el fichero.

- Para dar soporte a las soluciones voraces, se sugiere incorporar al programa una función de ordenación (utilizando por ejemplo el algoritmo Quicksort u otro de los algoritmos de ordenación vistos en clase) que tome el vector dinámico anterior y lo ordene creciente o decrecientemente según el campo del struct que almacena el tiempo de proceso.

- Una vez ordenado el vector dinámico, basta con ir tomando los procesos de principio a fin (análogo al problema de la mochila visto en clase) y escribiendo en pantalla la secuencia de procesos y su tiempo de espera.