UNIVERSIDAD DE SANTIAGO DE CHILE

Algoritmos Avanzados
- Lab1 -
Fuerza Bruta

Cristhofer Parada

José Toro

12/12/2021

Introducción

A los estudiantes de la asignatura de Algoritmos Avanzados se les solicitó realizar la implementación en código, utilizando el lenguaje de programación C, de 2 problemas, el primero consiste en encontrar la suma mayor de un grupo de números, el cual dichos números están acompañados de una ponderación, por lo que en el primer caso se debe encontrar la suma mayor de números sin superar la suma de ponderación total dada por el nombre de archivo. El segundo problema consiste en realizar, en pocos términos, un recorrido de grafo y encontrar el camino que sea menor. Para realizar estos trabajos a los estudiantes se les dio plena libertad para intentar resolver los problemas de manera libre, haciendo enfoque en que el programa utilice fuerza bruta para poder encontrar la solución al problema. En este caso, para resolver el problema se utilizó el método de Búsqueda en Espacio de Estados, el cual se basa en fuerza bruta y apunta a realizar todas las soluciones posibles dado un problema.

Método

Para resolver los problemas planteados se utilizó Búsqueda en Espacio de Estados (búsqueda en anchura), con este enfoque se logran generar todas las posibles soluciones a un problema, de esta forma encontraremos si o si la mejor solución, sin embargo, el crear tantas posibles soluciones tiene un tiempo que aumenta considerablemente dependiendo del problema a resolver, la estructura que obtiene el código al tener este enfoque es bastante simple, se generan todas las soluciones posibles utilizando estructuras, las cuales se generan mediante iteraciones, generando así distintos "estados" del problema, estos estados son generados realizando distintos cálculos, por ejemplo, para el problema de los grafos, se generan distintos estados iniciales, los cuales empiezan en un punto del grafo, por ejemplos en el "nodo 1", luego se genera el siguiente estado, el cual "avanza" hasta el siguiente nodo, general así un estado nuevo el cual tiene el recorrido [1,2], luego se genera el siguiente estado, utilizando el estado inicial como base generando el nuevo estado con un recorrido [1,3], de esta forma generamos todos los posibles estados.

Este algoritmo fue el elegido debido a su gran beneficio de generar todas las posibles soluciones, sin embargo, se presentó un problema lo cual fue el tiempo de ejecución debido a la gran cantidad de soluciones que son generadas, este problema queda en evidencia al realizar la ejecución del primer problema donde el primer documento es resuelto en cuestión de minutos, sin embargo, el resto tarda horas, incluso días, aunque este problema se tenía contemplado al momento de diseñar la posible solución de los problemas.

Discusión

A continuación, se mostrarán los resultados obtenidos a partir de la implementación realizada para cada problema.

En primer lugar, se comenzará analizando los resultados del algoritmo N°1, el cual consistía de los casos prueba que se pueden observar en la Figura 1.

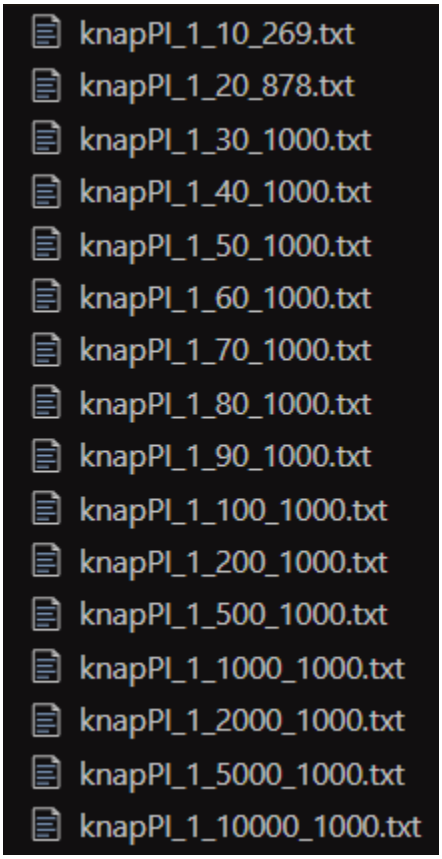

Figura 1

Dada la naturaleza del problema y la cantidad de soluciones que se pueden lograr a generar, las pruebas iniciales se realizaron todas utilizando como archivo de entrada el llamado “knapPI_1_10_269.txt”. Con dicho archivo de entrada, se crearon aproximadamente cien mil soluciones distintas, de las cuales se tuvo que buscar la que calce con el pMax, que en este caso corresponde a 269, y que su suma de valores sea la mayor de todas, la solución encontrada y su ejecución puede verse a continuación:

```
C:\Users\Harunomi\Desktop\Programas pencas\Algoritmos Avanzados\Lab1\Algoritmos_Avanzados_2-2021\Lab1>1
Ingrese el nombre del archivo a leer (con su respectiva extension)
por ejemplo knapPI_1_10_269.txt
knapPI_1_10_269.txt

Espere un momento estamos generando la solucion
posibles soluciones  10000
posibles soluciones  20000
posibles soluciones  30000
posibles soluciones  40000
posibles soluciones  50000
posibles soluciones  60000
posibles soluciones  70000
posibles soluciones  80000
posibles soluciones  90000
posibles soluciones  100000
Buscando la mejor solucion...

Solucion encontrada

ID: 18444
ID ANTERIOR: 3627
P ACTUAL: 269
MAXIMO ACTUAL: 295
```

Figura 2

Como podemos observar, la solución mayor es la 18444 de un total de aproximadamente cien mil, lo cual esto nos puede indicar que realizamos demasiadas soluciones que probablemente sean repetidas, lo cual claramente se puede mejorar realizando un mejor filtro, pero como dentro de los requisitos del problema fue solicitado que se utilice fuerza bruta de forma que el programa tuviese un mal tiempo de ejecución, se intentó realizar el menor tipo de optimización posible. Dicho lo anterior, al intentar ocupar otro archivo de entrada, como por ejemplo “knaPI_1_20_878.txt”, el tiempo de ejecución del programa crece de tal forma que incluso luego de 24 horas de ejecución (como se puede ver en la sección de apéndice, continúa realizando soluciones, por lo que se decidió que claramente aquello no estaba correcto y el propósito original del problema fue logrado, ya que si bien es cierto no se llegó a ningún resultado, se logró comprobar que utilizando fuerza bruta para resolver ciertos problemas no es la manera más eficiente y podemos encontrarnos con situaciones como esta en la cual el programa posiblemente necesitará días para poder dar con la solución correcta.

Por otro lado, haciendo un análisis del algoritmo N°2, acá se puede ver mejores resultados, dado que dentro de los archivos de entrada encontramos que el problema apunta a que debemos

llegar a un objetivo, es decir, se sabe el camino total que requiere el grafo para su recorrido. Esto disminuye en gran medida el tiempo de ejecución del programa, porque puede que haya múltiples soluciones que lleguen al resultado esperado, pero una vez encontramos una, el programa puede detenerse. Los resultados de dicho algoritmo pueden verse a continuación:

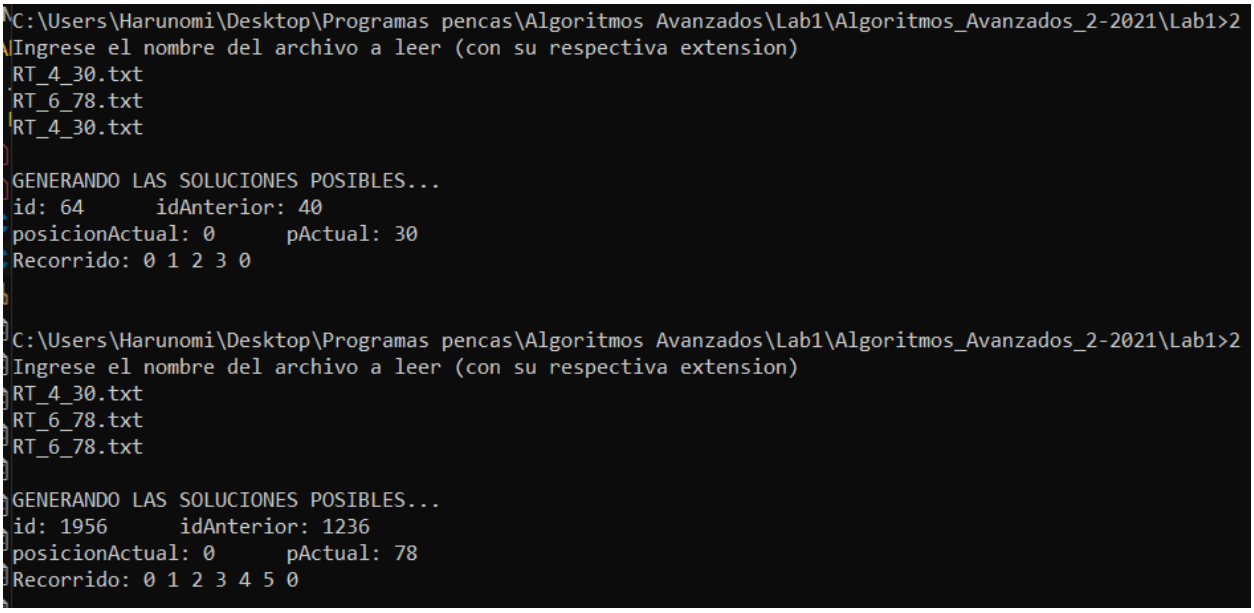

Figura 3

Los resultados anteriores, fueron obtenidos de manera inmediata, gracias a la naturaleza del problema y el hecho de saber al resultado a cual se espera llegar, por lo que a diferencia de lo logrado en el algoritmo N°1 podemos decir que este está mucho mejor optimizado y se logran encontrar las soluciones en ambos casos de ejemplo, pero a su vez esto viene dado que recorrer un grafo no supone realizar tantas iteraciones, dada la cantidad de nodos, en comparación con el algoritmo N°1 que posee un conjunto numérico mucho mayor.

Conclusiones

Los problemas planteados si bien son distintos, tras ser analizados con detenimiento se resuelven de manera muy similar, gracias a esto el tiempo empleado en crear las soluciones es bastante bajo debido a la simpleza del código diseñado, sin embargo, que sea simple no lo convierte en un código eficiente, por el contrario, ya que se generan todas las soluciones posibles se utiliza una gran cantidad de tiempo, esto queda en evidencia en el apéndice (Figura 4 y Figura 5) en las que existe un aproximado de 3 horas de diferencia ,sin embargo, solo se generaron cerca 110.000 nuevas soluciones, a pesar de este problema el objetivo se cumple, ya que se encuentran las mejores soluciones para ambos tipos de problemas (Figura 2 y Figura 3).

Apéndice(opcional)

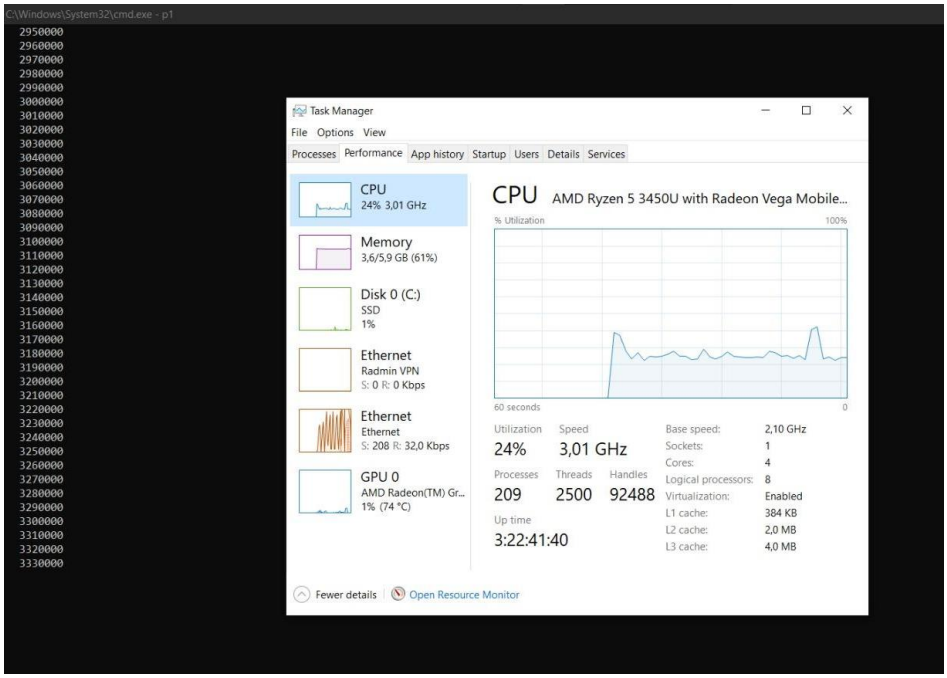

Figura 4: Ejecución problema 1 - archivo “knapPI_1_20_878.txt” - Tiempo 3:22:41

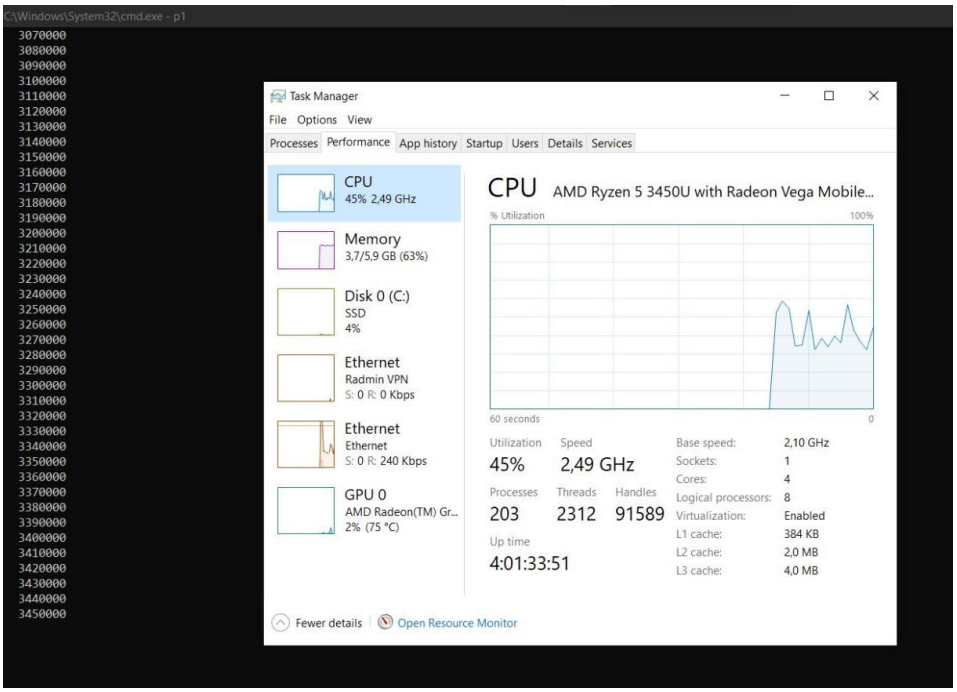

Figura 5: Ejecución problema 1 - archivo “knapPI_1_20_878.txt” - Tiempo 4:01:33

Referencias

Para resolver estos problemas no se utilizó ningún tipo de documentación aparte de los conocimientos obtenidos en cursos previos como es el caso de la Búsqueda en Espacio de Estados (BEE) la cual se vio en el curso de “Métodos de Programación”.