08112 Algorithms Coursework

Algorithm code

public static void Search1(double[] data, out int bestStart, out int bestEnd, out double bestTotal, out int loops)

{

bestTotal = 0;

bestStart = 0;

bestEnd = 0;

loops = 0;

double subtotal;

/// TODO - put your search code here

for (int i = 0; i < data.Length; i++)

{

for (int j = 0; j < data.Length; j++)

{

subtotal = 0;

for (int k = i; k <= j; k++)

{

loops++;

subtotal += data[k];

}

if (subtotal > bestTotal)

{

bestTotal = subtotal;

bestStart = i;

bestEnd = j;

}

}

}

}

public static void Search2(double[] data, out int bestStart, out int bestEnd, out double bestTotal, out int loops)

{

bestTotal = 0;

bestStart = 0;

bestEnd = 0;

loops = 0;

double subtotal;

/// TODO - put your search code here

for (int i = 0; i < data.Length; i++)

{

subtotal = 0;

for (int j = i; j < data.Length; j++)

{

loops++;

subtotal += data[j];

if (subtotal > bestTotal)

{

bestTotal = subtotal;

bestStart = i;

bestEnd = j;

}

}

}

}

public static void Search3(double[] data, out int bestStart, out int bestEnd, out double bestTotal, out int loops)

{

bestTotal = 0;

bestStart = 0;

bestEnd = 0;

loops = 0;

double subtotal;

int start = 0;

/// TODO - put your search code here

subtotal = 0;

for (int i = 0; i < data.Length; i++)

{

loops++;

subtotal += data[i];

if (subtotal > bestTotal)

{

bestTotal = subtotal;

bestStart = start;

bestEnd = i;

}

if (subtotal < 0)

{

start = i + 1;

subtotal = 0;

}

}

}

}

Data Tables

Program Outputs

|  |  |  |  |
| --- | --- | --- | --- |
| Data - 52 | Total | Start | End |
| Algorithm 1 | 99.72 | 17 | 20 |
| Algorithm 2 | 99.72 | 17 | 20 |
| Algorithm 3 | 99.72 | 17 | 20 |

|  |  |  |  |
| --- | --- | --- | --- |
| Data - 104 | Total | Start | End |
| Algorithm 1 | 167.01 | 77 | 98 |
| Algorithm 2 | 167.01 | 77 | 98 |
| Algorithm 3 | 167.01 | 77 | 98 |

|  |  |  |  |
| --- | --- | --- | --- |
| Data - 208 | Total | Start | End |
| Algorithm 1 | 366.43 | 122 | 207 |
| Algorithm 2 | 366.43 | 122 | 207 |
| Algorithm 3 | 366.43 | 122 | 207 |

Loops

|  |  |  |  |
| --- | --- | --- | --- |
|  | Algorithm 1 | Algorithm 2 | Algorithm 3 |
| Data - 52 | 24804 | 1378 | 52 |
| Data - 104 | 192920 | 5460 | 104 |
| Data - 208 | 1521520 | 21736 | 208 |

O(n) notation process

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Data Vol | Ratio | Alg 1 | Ratio | Alg 2 | Ratio | Alg 3 | Ratio |
| 52 |  | 24804 |  | 1378 |  | 52 |  |
| 104 | 2 | 192920 | 7.78 | 5460 | 3.96 | 104 | 2 |
| 208 | 2 | 1521520 | 7.87 | 21736 | 3.98 | 208 | 2 |
| Limit | 2 |  | 8 |  | 4 |  | 2 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Data Vol | Ratio | Alg 1 | Ratio | Alg 2 | Ratio | Alg 3 | Ratio |
| Limit | 2 | O(n^3) | 8 | O(n^2) | 4 | O(n) | 2 |

Write up

The big “O” notation is a way to show a programs efficiency and there are many different forms of the notation all depicting a different efficiency curve with time increasing as the number of inputs increase. One of the most efficient forms is O(n) where the time increases linearly with the number of inputs, the least efficient is O(n!) where the time increases a massive amount when more inputs are added.

To find the big “O” notation of each algorithm requires a few things. First the algorithms must be coherent with each other in terms of the outputted results from the algorithms being run on increasing size of files, starting from 52 items to 104 items then 208. This can be seen by the first 3 table’s show that all the algorithms find the same optimal results for each file when run.

The next table shows the amount of loops each algorithm goes through with each file, the number of loops is what is used to compare each algorithm in terms of efficiency eg. The higher the number of loops the less efficient the algorithm is. Looking at the table you can see that algorithm 1 is the least efficient and algorithm 3 is the most efficient.

Using these values you can figure out the ratios required to find the limits needed to figure out the big “O” notation as seen in the second to last table. By dividing the second number by the first and the third by the second you can figure out the ratios of each process and using the 2 ratios found you can approximate the limit as the nearest whole number that it will never reach as you increase the data vol.

Using the limits found you can find the big “O” notation of the algorithms by comparing the limits to the limits on the data vol eg. For algorithm 1 the limit is 8 when compared to the original 2 you know that 2^3 = 8 so the big “O” notation for algorithm 1 is O(n^3). The same process can be used for algorithm 2 and 3 giving the results shown in the last table.