KLE Technological University
Creating Value
Leveraging Knowledge

Algorithmic Problem Solving [23ECSE309]

Semester: VI **2023-24**

Q-Box Assignment Set

Student Name: Poonam M Shettar
SRN: 01FE21BEC280
Branch: Electronics and Communication

Question 01

Title: Stock ticker symbol identifier and price gain analysis
Level: Easy
Concepts Tested: maximum subarray sum, encoding, max and min in max subarray
Algorithm used: Kadane

Problem Statement:
Develop a real-time financial monitoring system for a stock trading platform where stock ticker symbols, represented in capital alphabets, need to be mapped to numerical identifiers ranging from 10 to 36(A-10 Z-36). The system must efficiently identify the subperiod with the maximum price gain to determine the most profitable period. It should pinpoint the start and end points of this maximum price range period and calculate both the minimum and maximum values from the price gain array within this subarray.

Input Format:
The first line contains a string representing stock ticker symbols, all in uppercase.
The second line contains an array of integers representing the price gains over several instances of time.

Constraints:

- Stock ticker symbols consist only of uppercase letters ('A'-'Z').
- Number of elements in the price gain array ranges from 0 to 100000.

Output Format:
First line contains the stock ticker identifier (with no space).
Second line contains the maximum price gain.
Third line contains the index of instances for start and end of maximum price gain subperiod.
Fourth line contains the maximum value the maximum price gain period.
Fifth line contains the minimum value in the maximum price gain period.

Solution:

```
#include <bits/stdc++.h>
using namespace std;

void encode(string stock_t)
{
  for (int i = 0; i < stock_t.length(); i++)
  {
    cout << int(stock_t[i] - 55); // stock ticker identifier
  }
  cout << endl;
}

void maximum_price_gain(vector<int> &price_changes)
{
  int sum = 0;
  int maxi = INT_MIN;
  int si = 0, ei = 0, temp_start = 0;
  int maxi_ = INT_MIN, mini_ = INT_MAX;
  int max_, min_;

  for (int i = 0; i < price_changes.size(); i++)
  {
    sum += price_changes[i];

    if (sum > maxi)
    {
      maxi = sum;
      si = temp_start;
      ei = i;
      max_ = maxi_;
      min_ = mini_;
    }

    if (sum < 0)
    {
      sum = 0;
      temp_start = i + 1;
      maxi_ = INT_MIN;
      mini_ = INT_MAX;
    }
```

```
      maxi_ = max(maxi_, price_changes[i]);
      mini_ = min(mini_, price_changes[i]);
    }

    cout << maxi << endl; //"Maximum Price Gain: "
    cout << si << '\t';   //"Start Index: "
    cout << ei << endl;   //"End Index: "
    cout << max_ << endl; //"Maximum Value in Max Price gain period: "
    cout << min_ << endl; // "Minimum Value in Max Price gain period: "
}

int main()
{
    string stock_ticker;
    vector<int> price_changes(5);

    // cout << "Enter stock ticker: ";
    cin >> stock_ticker;

    // cout << "Enter 5 price changes: ";
    for (int i = 0; i < 5; i++)
    {
        cin >> price_changes[i];
    }

    encode(stock_ticker);
    maximum_price_gain(price_changes);

    return 0;
}
```

Sample Test Cases:

Input

AAP

1 2 3 -1 4

Output

101025

9

0 4

3

-1

Input
MSFT
2 1 -1 3 4 5
Output
22281529
9
0 4
3
-1
Test Cases:
Attached in files.

Question 02

Title: Sentence Extraction Tool for Text Analysis
Level: Medium
Concepts Tested: Pattern matching
Algorithm used: Knuth-Morris-Pratt (KMP)

Problem Statement:
Develop a text analysis tool designed to process diverse text files encompassing articles, reports, logs, or other forms of content. These files are repositories of essential information. The objective is to implement a function capable of systematically parsing these files to identify predefined patterns or keywords. For each identified instance, the tool should extract and return the complete sentence containing the pattern, delimited by one full stop to the next.

Input Format:
A single text file containing the textual content to be analysed and a pattern.
Constraints:
It should be capable of processing extensive amounts of text data within reasonable time and memory constraints.
Output Format:
All sentences with matched keywords in new lines.
Print "NULL" if no match found.

Solution:

```
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
using namespace std;
vector<int> computeLPS(const string &pattern)
```

```
{
  int n = pattern.size();
  vector<int> lps(n, 0);
  int len = 0;
  int i = 1;

  while (i < n)
  {
    if (pattern[i] == pattern[len])
    {
      len++;
      lps[i] = len;
      i++;
    }
    else
    {
      if (len != 0)
      {
        len = lps[len - 1];
      }
      else
      {
        lps[i] = 0;
        i++;
      }
    }
  }
  return lps;
}

vector<int> KMPSearch(const string &text, const string &pattern)
{
  int n = text.size();
  int m = pattern.size();
  vector<int> lps = computeLPS(pattern);
  vector<int> indices;
  int i = 0; // index for text[]
  int j = 0; // index for pattern[]

  while (i < n)
  {
    if (pattern[j] == text[i])
```

```
        {
            j++;
            i++;
        }

        if (j == m)
        {
            indices.push_back(i - j); // Found pattern at index i-j
            j = lps[j - 1];
        }
        else if (i < n && pattern[j] != text[i])
        {
            if (j != 0)
            {
                j = lps[j - 1];
            }
            else
            {
                i = i + 1;
            }
        }
    }
    return indices;
}

string readTextFromFile(const string &filename)
{
    ifstream infile(filename);
    string line, content;

    while (getline(infile, line))
    {
        content += line + " ";
    }

    infile.close();
    return content;
}

vector<string> readPatternsFromFile(const string &filename)
{
    ifstream infile(filename);
```

```
    string line;
    vector<string> patterns;

    while (getline(infile, line))
    {
        patterns.push_back(line);
    }

    infile.close();
    return patterns;
}

void findSentences(const string &content, const vector<string> &keywords)
{
    for (const auto &keyword : keywords)
    {
        vector<int> indices = KMPSearch(content, keyword);
        if (indices.empty())
        {
            cout << "Keyword: " << keyword << endl;
            cout << "NULL" << endl
                 << endl;
        }
        else
        {
            cout << "Keyword: " << keyword << endl;
            for (int index : indices)
            {
                int start = content.rfind('.', index);
                int end = content.find('.', index + keyword.size());
                if (start == string::npos)
                    start = -1;
                if (end == string::npos)
                    end = content.size();

                string sentence = content.substr(start + 1, end - start);
                cout << sentence << endl;
            }
            cout << endl;
        }
    }
}
```

```
int main()
{
   string textFilename = "test_cases\\testcaseo.txt";
   string patternFilename = "test_cases\\testcaseo_pattern.txt";

   string content = readTextFromFile(textFilename);
   vector<string> keywords = readPatternsFromFile(patternFilename);

   findSentences(content, keywords);

   return o;
}
```

Sample Test Case:
Input:
(text)The quick brown fox jumps over the lazy dog. This is an example sentence. Another sentence follows.
machine learning has revolutionized many fields. Deep learning is a subset of machine learning.
Climate change is a pressing issue. It affects global weather patterns and sea levels.

(pattern)machine learning

Output:
machine learning has revolutionized many fields.
Deep learning is a subset of machine learning.

Test Cases:
Attached in files.