CS1020E Tutorial + Lab 07

Mark NG

a0116298@u.nus.edu
http://mollymr305.github.io

October 14, 2016

Tutorial Solutions

"Tutorial 8 – Complexity Analysis"

Question 1: Big-O Analysis

Rearrange the 15 terms in ascending order of their Big-O time complexity:

Question 1: Big-O Analysis

Rearrange the 15 terms in ascending order of their Big-O time complexity:

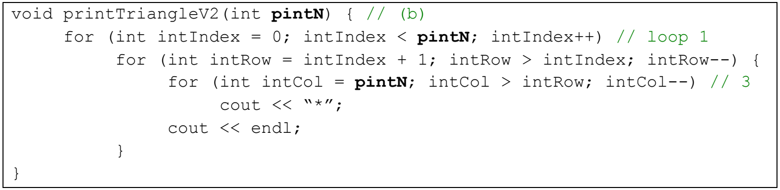
$4N^2$, $\log_5(N)$, $20N$, $N^{2.5}$, $\log(N!)$, N^N, 3^N, $N\log(N)$, $100N^{2/3}$, $\log(N)$, 2^N, 2^{N+1}, $N!$, $(N-1)!$, 2^{2N}

Question 2: Analysis of Iterative Algorithms

```
void printTriangle(int pintN) { // (a)
     for (int intRow = 0; intRow < pintN; intRow++) { // loop 1
          for (int intCol = intRow; intCol < pintN; intCol++) // loop 2
               cout << “*”;
          cout << endl;
     }
}
```

Figure : Question 2 (a)

Question 2: Analysis of Iterative Algorithms



```
void printTriangleV2(int pintN) { // (b)
     for (int intIndex = 0; intIndex < pintN; intIndex++) // loop 1
          for (int intRow = intIndex + 1; intRow > intIndex; intRow--) {
               for (int intCol = pintN; intCol > intRow; intCol--) // 3
                    cout << "*";
               cout << endl;
          }
}
```

Figure : Question 2 (b)

Question 2: Analysis of Iterative Algorithms

```
void clear(vector<int>& items) { // (c)
     int intN = items.size();

     for (int intIndex = 0; intIndex < intN; intIndex++) // loop 1
          items.erase(items.begin());
}
```

Figure : Question 2 (c)

Question 2: Analysis of Iterative Algorithms

```
void clear(vector<int>& items) { // (d)
     int intN = items.size();
     for (; --intN >= 0;) items.erase(items.begin() + intN); // loop 1
}
```

Figure : Question 2 (d)

Question 2: Analysis of Iterative Algorithms

```
void mystery(list<int>& items) { // (e)
     int intN = items.size() / 2;
     while (intN > 0) { // loop 1
          list<int>::iterator itr = items.begin();
          advance(itr, intN); // move forward N elements
          cout << *itr << " ";
          intN /= 2;
     }
}
```

Figure : Question 2 (e)

Question 2: Analysis of Iterative Algorithms

```
void guessWhatThisIs(vector<int>& items) { // (f)
     int intN = items.size();
     for (int intEnd = intN - 1; intEnd > 0; intEnd--) // loop 1
          for (int intLeft = 0; intLeft < intEnd; intLeft++) // loop 2
               if (items.at(intLeft + 1) < items.at(intLeft)) {
                    int intTemp = items.at(intLeft);
                    items.at(intLeft) = items.at(intLeft + 1);
                    items.at(intLeft + 1) = intTemp;
               }
}
```

Figure : Question 2 (f)

Question 3: Analysis of Recursive Algorithms

```
long long power(long long x, long long k, long long M) {
    if (k == 0) return 1;
    long long y = k / 2;
    if (2 * y == k) { // even power k
        long long half = power(x, y, M); // (x^y) % M
        return half * half % M; // [(x^y % M)(x^y % M)] % M
    } else { // k == 2y + 1
        long long next = power(x, 2 * y, M); // (x^2y) % M
        return x * next % M; // [x (x^2y % M)] % M
    }
}
```

Figure : Question 3: Sample

Question 3: Analysis of Recursive Algorithms

```
long long powerSum(long long x, long long k, long long M) { // (a)
    if (k == 1) return x % M;
    long long y = k / 2;
    if (k % 2 == 0) { // even power k
        long long half = powerSum(x, y, M); // S(y) % M
        long long pw = power(x, y, M); // (x^y) % M
        long long ans = half * (pw + 1); // (S(y) % M) [1 + (x^y % M)]
        return ans % M;
    } else { // k == 2y + 1
        long long next = powerSum(x, y + y, M); // S(2y) % M
        long long pw = power(x, k, M); // x^(2y + 1) % M
        long long ans = next + pw; // (S(2y) % M) + (x^(2y + 1) % M)
        return ans % M;
    }
}
```

Figure : Question 3: (a)

Question 3: Analysis of Recursive Algorithms

```
bool lookHere(vector<int>& items, int value, int low, int hi); // (b)
bool lookHere(vector<int>& items, int value) {
     int intN = items.size() - 1;
     return lookHere(items, value, 0, intN);
}
bool lookHere(vector<int>& items, int value, int low, int hi) {
     if (low > hi) return false;
     int mid = (low + hi) / 2;
     // do some O(1) stuff
     if (items.at(mid) > value)
          return lookHere(items, value, low, mid - 1);
     else
          return lookHere(items, value, mid + 1, hi);
}
```

Figure : Question 3: (b)

Question 3: Analysis of Recursive Algorithms

```
void lookHere(vector<int>& items, int value, int low, int hi); // (c)
void lookHere(vector<int>& items, int value) {
     int intN = items.size() - 1;
     lookHere(items, value, 0, intN);
}
void lookHere(vector<int>& items, int value, int low, int hi) {
     if (low >= hi) return;
     int mid = (low + hi) / 2;
     // do some O(1) stuff
     lookHere(items, value, low, mid);
     lookHere(items, value, mid + 1, hi);
}
```

Figure : Question 3: (c)

Question 3: Analysis of Recursive Algorithms

```
void lookHere(vector<int>& items, int value, int low, int hi); // (d)
void lookHere(vector<int>& items, int value) {
     int intN = items.size() - 1;
     lookHere(items, value, 0, intN);
}
void lookHere(vector<int>& items, int value, int low, int hi) {
     if (low >= hi) return;
     int mid = (low + hi) / 2;
     // do some O(N) stuff
     lookHere(items, value, low, mid);
     lookHere(items, value, mid + 1, hi);
}
```

Figure : Question 3: (d)

Question 3: Analysis of Recursive Algorithms

```
void lookHere(vector<int>& items, int value, int low, int hi); // (e)
void lookHere(vector<int>& items, int value) {
     int intN = items.size() - 1;
     lookHere(items, value, 0, intN);
}
void lookHere(vector<int>& items, int value, int low, int hi) {
     if (low >= hi) return;
     int mid = (low + hi) / 2;
     // do some O(N^2) stuff
     lookHere(items, value, low, mid);
     lookHere(items, value, mid + 1, hi);
}
```

Figure : Question 3: (e)

End of Tutorial Discussion

Note: Detailed solutions (i.e. the file `T8_ans.pdf`) will be released soon at

`http://www.comp.nus.edu.sg/~stevenha/cs1020e.html`

Let's take a short break!

Take Home Lab

Some notes...

Take Home Lab

Some notes...

- Please remember to use recursion to solve each sub task. If no recursion, then zero marks.

Take Home Lab

Some notes...

- Please remember to use recursion to solve each sub task. If no recursion, then zero marks.
- This is a *combinatorial problem*.

Take Home Lab

Some notes...

- Please remember to use recursion to solve each sub task. If no recursion, then zero marks.
- This is a *combinatorial problem*.
- Somewhat ‘related’ example (in terms of thinking):

  Given a string “abc”, can you count and print all possible permutations with repeats? What about without repeats?

t

Another Problem

Write a `C++` program which takes in two strings A, B and prints:

- "Anagram" if A is an anagram of B.
- "Not Anagram" if A is not an anagram of B.

Kattis Problem

`https://open.kattis.com/problems/different`

Any Questions?

See you next week!