Homework 5

Nguyen Tien Duc - ITITIU18029

3.1

12

```c
#include <stdio.h>

int main() {
  int x = 1;
  int y = 2;
  int z = 3;
  printf("(%d, %d, %d)", x, y, z);

  int temp = x;
  x = y;
  y = z;
  z = temp;
  printf("(%d, %d, %d)", x, y, z);
}
```

⇒ At least 4 assignments are needed.

14

```c
#include <stdio.h>

int linear(int a[], int size, int key) {
  for (int i = 0; i < size; i++)
    if (a[i] == key)
      return i;
  return -1;
}

int binary(int a[], int left, int right, int key) {
  if (left > right)
    return -1;

  int mid = (left+right)/2;
  if (a[mid] > key)
    binary(a, left, mid - 1, key);
  else if (a[mid] < key)
    binary(a, mid + 1, right, key);
  else if (a[mid] == key)
    return mid;
}

int main() {
  int sequence[] = {1, 3, 4, 5, 6, 8, 9};

  int key = 7;

  // return -1 means not found
  printf("Linear search: %d\n", linear(sequence, 7, key));
  printf("Linear search: %d\n", binary(sequence, 0, 6, key));
}
```

- Linear search: For linear search, we simply compare the key to every value in the sequence. If the value can't be found, return -1.
- Binary search: In binary search, we compare the value at the middle index with the key, if the key is large, the value before the middle index becomes the new right bound. Reversely, if the key is smaller, the value after the middle index becomes the new left bound. We do those steps recurs

20

```
#include <iostream>
using namespace std;

int main () {
  int sequence[10]={2, 3, -12, -23, 57, 12,-9, 32, -23, -3};
  int max = 0;
  int min = 0;
  int max_index = 0;
  int min_index = 0;

  for (int i = 0; i < 10; i++) {
    if (max < sequence[i]) {
      max = sequence[i];
      max_index = i;
    }
    if (min > sequence[i]) {
      min = sequence[i];
      min_index = i;
    }
  }

    cout << "Max is " << max << " at index " << max_index << endl;
    cout << "Min is " << min << " at index " << min_index << endl;
}
```

22

We first scan the sentence into a string.
Then we loop through it, adding each word, separated by spaces into a vector of strings (each string is a word).
Then, we can use the vector of words to scan for the longest word.

```
#include <iostream>
#include <vector>
using namespace std;

int main() {
  string longest_word;
  int max_length = 0;
  string sentence;
  vector<string> set_of_words;

  puts("Input an English sentence.");
  getline(cin, sentence);

  string word_tmp = "";
  for (string::iterator i = sentence.begin(); i <= sentence.end(); i++) {
    if (*i != ' ' && i != sentence.end()) {
      word_tmp.push_back(*i);
    }
    else {
      cout << word_tmp << endl;
      set_of_words.push_back(word_tmp);
      word_tmp = "";
    }
  }

  for (int i = 0; i < set_of_words.size(); i++) {
    if (max_length < set_of_words[i].size()) {
      max_length = set_of_words[i].size();
      longest_word = set_of_words[i];
    }
  }

  cout << max_length << endl;
  cout << longest_word << endl;
}
```

3.3

2

```
#include <iostream>
using namespace std;

int main() {
  int a[4] = {4, 3, 2, 1};
  for (int i = 0; i < 3; i++)
    if (a[i] > a[i+1]) {
      int temp = a[i];
      a[i] = a[i+1];
      a[i+1] = temp;
    }
}
```

As the code above, we use 2 loops to compare each number with all of its successive numbers. Therefore:

- The first term is compared with 3 successive terms.
- The second is compared with 2 successive terms.
- The third is only compared with 1 successive term.

In total, there are 6 comparisons made, and as we only need to sort the first 4 terms of an arbitrary list of numbers, the size of the list does not matter. Then the complexity is O(1).

4

If we successively square x to obtain x^{2^k}, then we will need to square x k times. Thus, require k multiplications.
If we multiply x by itself, we will need to do 2^k times. Thus, require 2^k multiplications.
When k ¿ 1, then $2^k > k$ and we will require less multiplications by successively squaring x. This means that successively squaring x is more efficient than multiplying x by itself.

10

a) $T = 10^{-9}(2(10)^2 + 2^(10)) = 1224$ nanoseonds

b) $T = 10^{-9}(2(20)^2 + 2^(20)) = 1049376$ nanoseconds

c) $T = 10^{-9}(2(50)^2 + 2^(50)) = 13$ days

d) $T = 10^{-9}(2(100)^2 + 2^(100)) = 40.2$ trillion years

12

a) The algorithm only makes a comparison if max ¡ a_i. Thus, only one comparison is made per iteration. There are n-1 iterations (2 to n). Thus, in the best case, there are at least n-1 comparisons. The best case performance is O(n).

b) In the best case, the key will match at the end where linear search begin from. Thus, only one comparison will be made. The performance will be O(1.)

c) In binary search, the list is cut in half every iteration, the algorithm does not stop until the list has been cut til one element. Thus, the best case performance is still not much different from the worst case. Then the performance will be O(log n)

3.4

10

a) Quotient: 5 — Remainder:4

b) Quotient: 37 — Remainder: 0

c) Quotient: -7 — Remainder: 10

d) Quotient: -1 — Remainder: 22

e) Quotient: -24 — Remainder: 86

f) Quotient: 0 — Remainder: 0

g) Quotient: 1233 — Remainder: 334

h) Quotient: -1 — Remainder: 1

12

Lets: $a = mi + r$ and $b = mj + r'$
As $a\, mod\, m = b\, mod\, m$ then $r = r'$
Then, substitute $r = a - mi$ into r', we have $b = mj + a - mi$
$\Rightarrow b = m(j - i) + a$ or $a = b + (i - j)m$
$\Leftrightarrow a \equiv b\,(mod\, m)$

3.5

16

Mersenne's claims:

$$2^p - 1$$

is prime when p is ∈ 2, 3, 5, 7, 13, 17, 19, 31, 67, 127, 257.

a)
$$2^7 - 1 = 127 = 7 \times 73 \Rightarrow \text{Prime}$$

This confirms Mersenne's claim.

b)
$$2^9 - 1 = 511 \Rightarrow \text{Not Prime}$$

This confirms Mersenne's claim.

c)
$$2^11 - 1 = 2047 = 23 \times 89 \Rightarrow \text{Not Prime}$$

This confirms Mersenne's claim.

d)
$$2^13 - 1 = 8191 \Rightarrow \text{Prime}$$

This confirms Mersenne's claim.

20

a) $2^2 \cdot 3^3 \cdot 5^2 = 2700$

b) $2 \cdot 3 \cdot 11 = 66$

c) 17

d) 1

e) 5

f) $2 \cdot 3 \cdot 5 \cdot 7 = 210$

3.7

20

$$\begin{aligned} x &\equiv 5\,(mod\,6) && (1)\\ x &\equiv 3\,(mod\,10) && (2)\\ x &\equiv 8\,(mod\,15) && (3) \end{aligned}$$

We can simplify so that the moduli are pairwise relatively prime.

$$\begin{aligned} x &\equiv 1\,(mod\,2) && (1)\\ x &\equiv 2\,(mod\,3) && (2)\\ x &\equiv 3\,(mod\,5) && (3) \end{aligned}$$

$$\begin{aligned}
\text{From (3): } & x \equiv 5i+3 \\
\text{Substitute in (2): } & 5i+3 \equiv 2\,(mod\,3) \\
& \Leftrightarrow i \equiv 1\,(mod\,3) \\
& \Rightarrow i = 3j+1 \\
& \Rightarrow x = 15j+18 \\
\Rightarrow \text{Substitute in (1): } & 15j+18 \equiv 1\,(mod\,2) \\
& \Rightarrow j \equiv 1\,(mod\,2) \\
& \Rightarrow j = 2k+1 \\
\text{Now, we substitute back to i then to x:} & \\
& \Rightarrow i = 3\cdot(2k+1) = 6k+4 \\
& \Rightarrow x = 5\cdot(6k+4)+3 = 30k+23
\end{aligned}$$

Then, $x = 30j+23$, provided that j is an integer.

22

Suppose there are a and b with $0 \le a, b < m$ that have the same n-tuples. Then:
$(a\,mod\,m_1, a\,mod\,m_2, ...a\,mod\,m_n) = (b\,mod\,m_1, b\,mod\,m_2, ...b\,mod\,m_n)$
$\Rightarrow a \equiv b\,(mod\,m_i)(\forall i \in 1...n)$
By the Chinese remainder theorem:
$a \equiv b\,(mod\,m)$
a is congruent to b modulo m if m divides a -b
$m|(a-b)$
$Since\,0 \le a, b < m:\ -m < a-b < m$ and thus $m|(a-b)$ then implies $a-b=0$
Then, a = b. This contradicts with our previous assumption.
Then, a can be uniquely represented by the n-tuple

Calulate Summations and Products

1

$$\sum_{i=1}^{n}\sum_{j=1}^{m} i+j$$

```
#include <iostream>
using namespace std;

int main() {
  cout << "Input n and m: " << endl;
  int n, m;
  cin >> n;
  cin >> m;
  int sum = 0;
  for (int i = 0; i < n; i++)
    for (int j = 0; j < m; j++)
      sum = sum + i + j;
  cout << "Sum is " << sum;
}
```

2

$$\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{l}(i \cdot j) + k$$

```
#include <iostream>
using namespace std;

int main() {
  cout << "Input n and m and l: " << endl;
  int n, m, l;
  cin >> n;
  cin >> m;
  cin >> l;
  int sum = 0;
  for (int i = 0; i < n; i++)
    for (int j = 0; j < m; j++)
      for (int k = 0; k < l; k++)
         sum = sum + ((i + j)*k);
  cout << "Sum is " << sum;
}
```

3

$$\prod_{i=1}^{n}\prod_{j=1}^{m} i + j$$

```
#include <iostream>
using namespace std;

int main() {
  cout << "Input n and m: " << endl;
  int n, m;
  cin >> n;
  cin >> m;
  int product = 1;
  for (int i = 0; i < n; i++)
    for (int j = 0; j < m; j++)
      product = product*(i + j);
  cout << "Product is " << product;
}
```