Homework 4

Nguyễn Tiến Đức – ITITIU18029

2.

a) Page 280:

6.

- Base case: $n = 1$
  - ⇨ Left = $1 \cdot 1! = 1$
  - ⇨ Right = $(1+1)! - 1 = 2! - 1 = 1$
    - ⇨ True
- Inductive Hypothesis:
  - ⇨ Assume that for all integer $k > 0$:
    - ◆ $1 \cdot 1! + 2 \cdot 2! + \cdots + k \cdot k! = (k+1)! - 1$
- Inductive step:
  - ⇨ Then for $k + 1$:
    - ◆ Left = $1 \cdot 1! + 2 \cdot 2! + \cdots + k \cdot k! + (k+1)!\,(k+1)$

      = $P(k) + (k+1)!\,(k+1)$
    - ◆ Right = $(k+2)! - 1 = (k+2) \cdot (k+1)! - 1$

      = $\big((k+1)+1\big)(k+1)! - 1$

      = $(k+1)(k+1)! + (k+1)! - 1$

      = $(k+1)(k+1)! + P(k)$

      ⇨ Left = Right

❖ By induction, the equality is proved.

8.

- Base case: $n = 0$
  - ⇨ Left = $2 \cdot 7^0 = 2$
  - ⇨ Right = $\frac{1-(-7)^{0+1}}{4} = \frac{1-(-7)}{4} = \frac{8}{4} = 2$
    - ⇨ True
- Inductive Hypothesis:
  - ⇨ Assume that for all integer $k \geq 0$:
    - ◆ $P(k) = \sum_{0}^{\mathrm{k}} 2 \cdot (-7)^n = \frac{1-(-7)^{k+1}}{4}$
- Inductive step:

- ⇨ Then for $k+1$:
  - ◆ Left = $\sum_0^{k+1} 2 \cdot (-7)^n = 2 \cdot (-7)^{k+1} + \sum_0^k 2 \cdot (-7)^n$

    $= 2 \cdot (-7)^{k+1} + P(k)$
  - ◆ Right = $\frac{1-(-7)^{k+2}}{4} = \frac{1-(-7)^{k+1}\cdot(-7))}{4} = \frac{1+(-1+8)\cdot(-7)^{k+1}}{4}$

    $= \frac{1-(-7)^{k+1}+8\cdot(-7)^{k+1}}{4} = P(k) + 2 \cdot (-7)^{k+1}$

    ⇨ Left = Right

❖ By induction, the equality is proved.

12.

- Base case: $n = 0$
  - ⇨ Left = $\left(-\frac{1}{2}\right)^0 = 1$
  - ⇨ Right = $\frac{2^{0+1}+(-1)^0}{3\cdot 2^0} = \frac{2+1}{3} = 1$

    ⇨ True
- Inductive Hypothesis:
  - ⇨ Assume that for all integer $k \geq 0$:
  - ◆ $P(k) = \sum_0^k \left(-\frac{1}{2}\right)^j = \frac{2^{k+1}+(-1)^k}{3\cdot 2^k}$
- Inductive step:
  - ⇨ Then for $k+1$:
    - ◆ Left = $\sum_0^{k+1} \left(-\frac{1}{2}\right)^j = \left(-\frac{1}{2}\right)^{k+1} + P(k)$
    - ◆ Right = $\frac{2^{k+2}+(-1)^{k+1}}{3\cdot 2^{k+1}} = \frac{2^{k+1}\cdot 2+(-1)^k\cdot(-1)}{3\cdot 2^k \cdot 2} = \frac{2^{k+1}\cdot 2-(-1)^k}{3\cdot 2^k \cdot 2}$

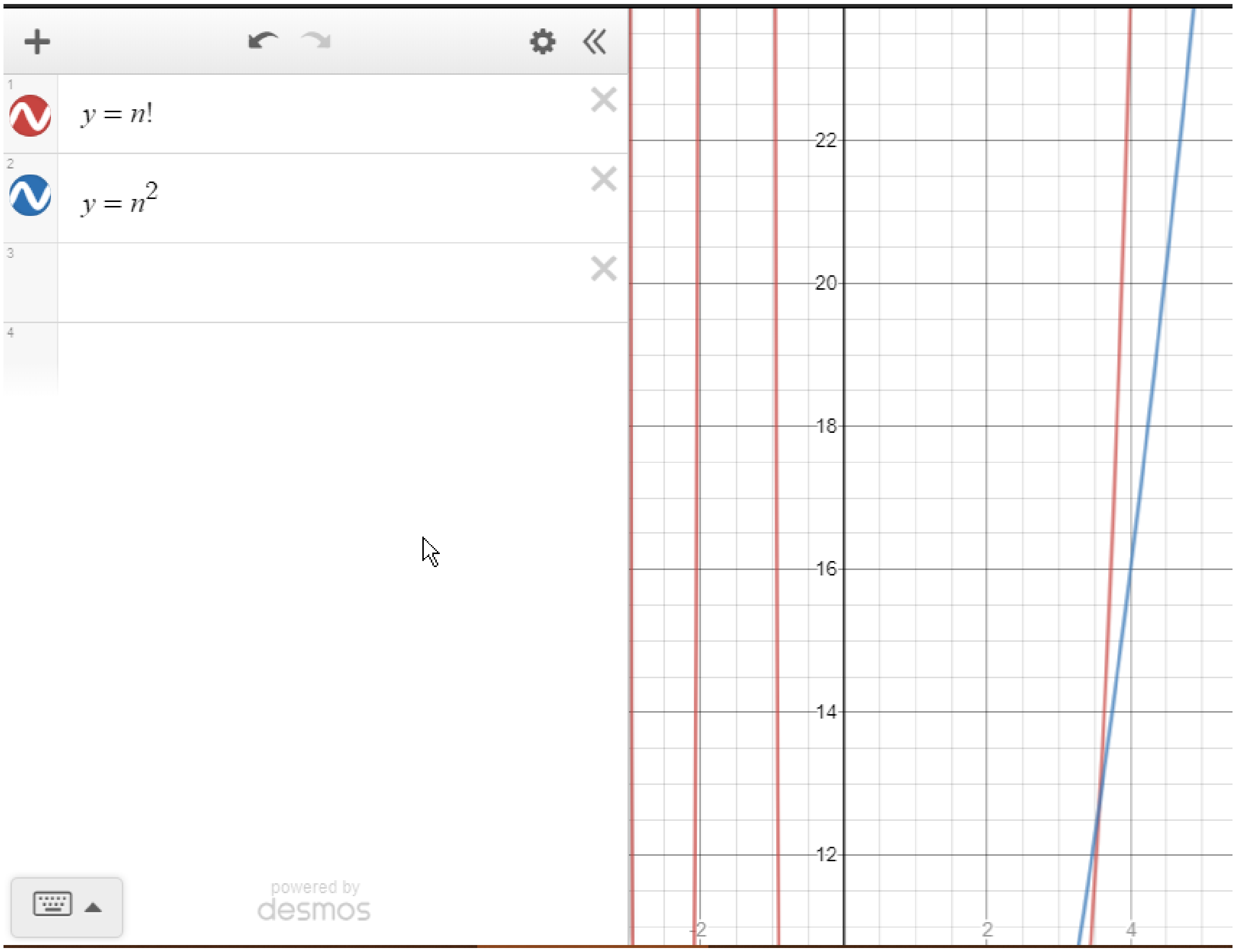
      $= \frac{2^{k+1}-\frac{1}{2}(-1)^k}{3\cdot 2^k} = \frac{2^{k+1}+(-1)^k-\frac{3}{2}(-1)^k}{3\cdot 2^k} = P(k) - \frac{(-1)^k}{2^{k+1}}$

      $= P(k) + \frac{(-1)\cdot(-1)^k}{2^{k+1}} = P(k) + \left(-\frac{1}{2}\right)^{k+1}$

      ⇨ Left = Right

❖ By induction, the equality is proved.

22.



By sketching the graph, we know for n ≥ 4 then $n^2 \le n!$

PROOF BY INDUCTION:

- Base case: $n = 4$
  - ⇨ Left = $4^2 = 16$
  - ⇨ Right = $4! = 24$
    - ⇨ Left < Right
    - ⇨ True
- Inductive Hypothesis:
  - ⇨ Assume that for all integer $k \ge 0$:
    - ◆ $P(k) = k^2 \le k!$
- Inductive step:
  - ⇨ Then for $k + 1$:
    - ◆ Left = $(k+1)^2 = k^2 + 2k + 1 = P(k) + 2k + 1$
    - ◆ Right =$(k+1)! = (k+1) \cdot k! = k \cdot k! + k!$

    Because $P(k) \le k!$, so if $2k + 1 \le k \cdot k!$, then it is proved:

    - ○ $k \cdot k! \ge k^3 > 2k + 1 \; \forall \; k \ge 4$
      - ⇨ Left ≤ Right

❖ By induction, the inequality is proved.

b) Page 291-292

6.

a) All combinations in the form: $3x + 10y$ $(x \geq 0; y \geq 0)$:

3, 6 ,9, 10, 12 ,13, 15, 16, 18, 19, 29, 21, 22, …

b) Prove for n ≥ 18

Basic step: n = 18 => 6 stamps of 3-cents.

Assume: A postage of k cents can be formed using just 3-cent and 10-cent stamps.

Inductive step: If k cents could be formed using 3 or more 3-cents stamps, then we can replace those 3 stamps with 1 10-cent stamps to have k+1 cent.

$$3 \cdot 3 + 1 = 10$$

If k cents could be formed using less than 3 3-cent stamps. Then the k cent was formed with at least 2 10-cent stamps (k≥18). Then, we replace 2 10-cent stamps with 7 3-cent stamps to obtain k+1 cent.

$$2 \cdot 10 + 1 = 3 \cdot 7$$

Then, by induction, it is true for n ≥ 18.

c) Basic step:

n = 18: 6 x 3-cent stamps

n = 19: 3 x 3-cent stamps and 1 x 10-cent stamp

n = 20: 2 x 10-cent stamps

Assume: A postage ranging from 18 to k cents can be formed using just 3-cent and 10-cent stamps.

Inductive step: If it is true from 18 to k cent then for k + 1 cent, we just add 1 x 3-cent stamp into k-2 cent (which is true as in our assumption).

Then, by strong induction, it is true for all positive integers n.

Difference in inductive hypothesis: In mathematical induction, only P(k) is assumed. In strong induction, P is assumed for values range from the base up to k.

8.

The possible total amount is the combination: $25x + 40y = 5(5x + 8y)$ $(x, y \geq 0)$:

Basic step: Let $n = 5x + 8y$. Then, total amount is 5n.

P(n): We can form 5n dollars in gift certificates using just 25-dollars and 40-dollars certificates.

We can form total amounts of the form 5n for all $n \geq 28$.

$5 = 5$, $8 = 8$, $10 = 5+5$, $13 = 8 + 5 + 5$, $15 = 3 \times 5$, $16 = 2 \times 8$, …., $24 = 3 \times 8$, $25 = 5 \times 5$, … $31 = 2 \times 8 + 3 \times 5$, $32 = 4 \times 8$

P(k) is true for k = 28, 29, 30, 31, 32

Assume: P(k) is true for all j if $28 \leq j \leq k$, $k \geq 32$

Inductive step: $P(k + 1)$ is also true as $k - 4 \geq 28$, which is true, so we can add 1x, $(k - 4 + 5 = k + 1)$ to $P(k - 4)$ to get $P(k + 1)$

14.

Let P(n) be "the sum of the products if $\frac{n(n-1)}{2}$"

Basic step: n = 2

We can only split the 2 stones into 2 piles of 1 stone each => product is 1.

$$\frac{n(n-1)}{2} = \frac{2(2-1)}{2} = 1$$

⇨ P(2) is true.

Inductive Hypothesis:

Assume that P(1), P(2), …, P(k) are all true.

Inductive step:

For P(k+1), we split it into a pile of m stones and pile of k + 1 – m stones.

Because P(m) is true ($m \leq k + 1$), the sum of the products to obtain the pile of m stones is $\frac{m(m-1)}{2}$

Because P(k + 1 – m) is also true, the sum of the products to obtain the pile is:

$$\frac{(k+1-m)\cdot(k+1-m-1)}{2} = \frac{(k+1-m)(k-m)}{2}$$

The sum of the product of the pile of k + 1 stones is then:

$$\frac{(k+1-m)(k-m)}{2} + \frac{m(m-1)}{2} + m(k+1-m)$$

$= \frac{(k+1-m)(k-m)+m(m-1)+2m(k+1-m)}{2}$

$= \frac{k^2+k}{2}$

$= \frac{k(k+1)}{2}$

$= \frac{k((k+1)-1)}{2}$

Thus P(k+1) is true, as the sum of the product is $\frac{k((k+1)-1)}{2}$

- ⇨ By strong induction, P(n) is true for all positive integers n.

c) Prove $3^n < n!$ when ever n is a positive integer and $n > 6$:

- Base case: $n = 7$
  - ⇨ $3^n = 3^7 = 2187$
  - ⇨ $n! = 7! = 5040$
    - ⇨ $3^7 < 7!$
- Inductive Hypothesis:
  - ⇨ Assume for all integer $k > 6$: $3^k < k!$
- Inductive step:
  - ⇨ Then for $k + 1$:
    - ◆ Left = $3^{(k+1)} = 3^k \cdot 3 < k! \cdot 3$
    - ◆ Right = $(k + 1)! = (k + 1) \cdot k!$
    - ◆ For all $k > 6$: $3 < k + 1$
      - ⇨ $k! \cdot 3 < (k + 1) \cdot k!$
      - ⇨ Left < Right

❖ By induction, the inequality is proved.

3 and 4.

Result:

```
Input number of terms: 30
Iteration(1) or Recursion(0): 0
Term Value
1    0
2    1
3    1
4    2
5    3
6    5
7    8
8    13
9    21
10   34
11   55
12   89
13   144
14   233
15   377
16   610
17   987
18   1597
19   2584
20   4181
21   6765
22   10946
23   17711
24   28657
25   46368
26   75025
27   121393
28   196418
29   317811
30   514229
```

```
Compare running time(1) or stop(0): 1
Term        Iteration       Recursion
1           1               3               microsecond
2           0               1               microsecond
3           1               1               microsecond
4           1               2               microsecond
5           1               2               microsecond
6           1               3               microsecond
7           1               6               microsecond
8           1               6               microsecond
9           1               8               microsecond
10          1               20              microsecond
11          1               30              microsecond
12          1               51              microsecond
13          1               77              microsecond
14          1               189             microsecond
15          1               201             microsecond
16          2               823             microsecond
17          1               958             microsecond
18          1               1874            microsecond
19          1               2370            microsecond
20          2               2431            microsecond
21          1               3429            microsecond
22          1               4806            microsecond
23          2               10023           microsecond
24          1               12912           microsecond
25          2               28825           microsecond
26          1               44083           microsecond
27          1               48090           microsecond
28          2               82252           microsecond
29          2               169885          microsecond
30          2               239729          microsecond
```

Source code:

```
#include <iostream>
#include <chrono>
#include <vector>
#include <iomanip>
using namespace std;
using namespace std::chrono;

enum Method { RECURSION, ITERATION };

unsigned int recursive_term(int t)
{
    if (t > 1)
```

```
        return recursive_term(t - 1) + recursive_term(t - 2);
    return t;
}

unsigned int iterative_term(int t)
{
    if (t > 1)
    {
        int next = 1;
        int now = 0;

        for (int i = 0; i < t; i++)
        {
            int temp = next;
            next += now;
            now = temp;
        }
        return now;
    }
    return t;
}

void printFibonacciList(int num, Method mtd)
{
    cout << left
        << setw(5) << "Term"
        << setw(15) << "Value"
        << endl;
    for (int i = 0; i < num; i++)
    {
        cout << setw(5) << i + 1;
        if (mtd == RECURSION)
            cout << setw(20) << recursive_term(i);
        else
            cout << setw(20) << iterative_term(i);
        cout << endl;
    }
}

auto TimeofTerm(int num, Method mtd)
{
    auto start = high_resolution_clock::now();
    if (mtd == RECURSION)
        recursive_term(num);
    else
        iterative_term(num);
    auto stop = high_resolution_clock::now();
    auto duration = duration_cast<microseconds>(stop - start);
    return duration.count();
}

void comparison(int num)
```

```
{
    cout << setw(10) << "Term"
        << setw(15) << "Iteration"
        << setw(15) << "Recursion"
        << endl;
    for (int i = 0; i < num; i++)
        cout << setw(10) << i + 1
            << setw(15) << TimeofTerm(i, ITERATION)
            << setw(15) << TimeofTerm(i, RECURSION)
            << "microsecond" << endl;
}

int main()
{
    int opt = 0;
    int num_of_term = 0;
    Method method;

    cout << "Input number of terms: ";
    cin >> num_of_term;

    cout << "Iteration(1) or Recursion(0): ";
    cin >> opt;

    if (opt == 1)
        method = ITERATION;
    else
        method = RECURSION;

    printFibonacciList(num_of_term, method);

    cout << endl << "Compare running time(1) or stop(0): ";
    cin >> opt;

    if (opt == 1)
        comparison(num_of_term);

    return 0;
}
```