Họ tên: Nguyễn Trung Dũng

MSSV: 19120486

Bài 1:

- Có:

$$f_1(n) \in O(g_1(n)) \Rightarrow f_1(n) \leq c_1.g_1(n) \forall n \geq n_1 (c_1 \in R; n_1 \in N)$$

$$f_2(n) \in O(g_2(n)) \Rightarrow f_2(n) \leq c_2.g_2(n) \forall n \geq n_2 (c_2 \in R; n_2 \in N)$$

$$\Rightarrow f_1(n) + f_2(n) \leq c_1.g_1(n) + c_2.g_2(n) \forall n \geq max(n_1, n_2)$$

- Đặt: $c = max(c_1, c_2)$, ta có:

$$f_1(n) + f_2(n) \leq c_1.g_1(n) + c_2.g_2(n) \leq c.\big(g_1(n) + g_2(n)\big)$$

- Mà: $a + b \leq 2max(a, b)$

$\Rightarrow\ c.\big(g_1(n) + g_2(n)\big) \leq 2c.max\{g_1(n), g_2(n)\}$

$\Rightarrow\ f_1(n) + f_2(n) \leq 2c.max\{g_1(n), g_2(n)\} \forall n \geq max(n_1, n_2)$

Vậy:

$$f_1(n) \in O(g_1(n)) \wedge f_2(n) \in O(g_2(n)) \Rightarrow f_1(n) + f_2(n) \in O(max\{g_1(n), g_2(n)\})$$

Bài 2:

- Chứng minh quy nạp: $\begin{bmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^n$ với $n \geq 1$
  - Đặt $A = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$
- Ta có dãy Fibonacci:

$$F_0 = 0$$

$$F_1 = 1$$

$$F_{n+1} = F_n + F_{n-1}$$

- Xét $n = 1$, ta thấy:

$$A^1 = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} F_2 & F_1 \\ F_1 & F_0 \end{bmatrix} \text{ (đúng)}$$

- Giả sử mệnh đề đúng với $n = k \geq 1$, tức là:

$$A^k = \begin{bmatrix} F_{k+1} & F_k \\ F_k & F_{k-1} \end{bmatrix}$$

Cần chứng minh mệnh đề đúng với $n = k + 1$, tức là cần chứng minh:

$$A^{k+1} = \begin{bmatrix} F_{k+2} & F_{k+1} \\ F_{k+1} & F_k \end{bmatrix}$$

Thật vậy:

$$A^{k+1} = AA^k = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} F_{k+1} & F_k \\ F_k & F_{k-1} \end{bmatrix} = \begin{bmatrix} F_{k+1} + F_k & F_k + F_{k-1} \\ F_{k+1} & F_k \end{bmatrix} = \begin{bmatrix} F_{k+2} & F_{k+1} \\ F_{k+1} & F_k \end{bmatrix}$$

Vậy:

$$\begin{bmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^n \; với\; n \geq 1$$

Bài 3:

- Thao tác cơ sở: `a[j - 1] > a[j]`
- Ta có:
  - *Trường hợp tốt nhất*: mảng được sắp xếp tăng dần, j chạy từ $n \to 2$. Vì vậy:

$$B(n) = n - 1 \in \Theta(n)$$

  - *Trường hợp xấu nhất*: mảng được sắp xếp giảm dần, ta có số lần so sánh cần thực hiện là:

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

$$\Rightarrow W(n) = \frac{n(n-1)}{2} \in \Theta(n^2)$$

  - *Trường hợp trung bình*:

    Tổng số lần thực hiện phép so sánh sau lần duyệt thứ i là:

$$\sum_{m=(n-i)}^{n-1} m = \frac{i(2n-i-1)}{2}$$

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

Mà xác suất để chương trình dừng tại mỗi lần duyệt là $\frac{1}{n-1}$ nên:

$$A(n) = \frac{1}{n-1}\sum_{i=1}^{n-1}\left(\frac{(2\mathrm{n}-1)\mathrm{i}}{2} - \frac{\mathrm{i}^2}{2}\right)$$

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

Mà:

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

$$\sum_{i=1}^{n-1} i^2 = \frac{(n-1)n(2n-1)}{6}$$

$$\Rightarrow A(n) = \frac{(2n-1)n}{4} - \frac{(2n-1)n}{12} = \frac{(2n-1)n}{6} \in \Theta(n^2)$$

Bài 4:

- Thao tác cơ sở: `a[j] > v`
- Ta có:
  - *Trường hợp tốt nhất*: i chạy từ $2 \to n$, `a[j] <= v` (mảng đã được sắp xếp không giảm) nên thao tác cơ sở chỉ được thực hiện 1 lần cho mỗi i

    $$B(n) = n - 1 \in \Theta(n)$$
  - *Trường hợp xấu nhất*: mảng đã được sắp xếp giảm dần, số lân tối đa thực hiện thao tác cơ sở cho mỗi i là $i - 1$ lần (j: $(i-1) \to 1$)

    $$W(n) = \sum_{i=2}^{n}(i-1) = \frac{n(n-1)}{2} \in \Theta(n^2)$$
  - *Trường hợp trung bình*:

    Xác suất để chèn phần tử thứ i vào i chỗ trong $i-1$ phần tử trước là $\frac{1}{i}$

    Số lần so sánh để chèn vào vị trí 1 là $i-1$ lần

Số lần so sánh để chèn vào vị trí $i - j + 1 \geq 2$ là j $(1 \leq j \leq i - 1)$ lần

Vậy số lần so sánh trung bình tại vòng lặp thứ i là:

$$C(i) = \frac{i-1}{i} + \frac{1}{i}\sum_{j=1}^{i-1} j = \frac{i^2 + 1 - 2}{2i} = \frac{i+1}{2} - \frac{1}{i}$$

$$\Rightarrow A(n) = \sum_{i=2}^{n} C(i) \approx \frac{n(n-1)}{4} + n - \ln n - \gamma \in \Theta(n^2)$$

Bài 5:

- Thao tác cơ sở: `curSum += a[k]`
- Xét vòng `for i:` $1 \rightarrow$ n
- Xét vòng `for j:` $i \rightarrow n$
  - Có thể thấy: hai vòng lặp đầu mỗi lần lặp sẽ xét một cặp (i, j) duy nhất $(1 \leq i \leq j \leq n)$; 2 vòng for này sẽ đi qua hết tất cả các cặp (i, j) như vậy
- Xét vòng `for k:` $i \rightarrow j$
  - Cứ mỗi cặp (i, j) thì k đi từ $i \rightarrow j$

    $\Rightarrow$ Thao tác cơ sở sẽ được lặp lại $j - i + 1$ lần đối với mỗi cặp (i, j)

Vậy, gọi $C(n)$ là tổng số lần thực hiện thao tác cơ sở với mảng n phần tử, ta có:

$$C(n) = \sum_{1 \leq \mathrm{i} \leq \mathrm{j} \leq \mathrm{n}} (\mathrm{j} - \mathrm{i} + 1)$$

- Xét các cặp (i, j) với chiều dài $m = j - i + 1$, số các cặp (i, j) như vậy trong khoảng $[1, n]$ là $n - (m - 1)$ cặp

  Vậy ta có:

$$C(n) = 1.n + 2(n-1) + 3(n-2) + \cdots + m\big(n - (m-1)\big) + \cdots + n.1$$

$$= \sum_{m=1}^{n} \big(m(n - m + 1)\big)$$

$$= (n+1)\sum_{m=1}^{n} m - \sum_{m=1}^{n} m^2$$

Mà

$$\sum_{m=1}^{n} m = \frac{\mathrm{n}(\mathrm{n}+1)}{2}$$

$$\sum_{m=1}^{n} m^2 = \frac{n(n+1)(2n+1)}{6}$$

$$\Rightarrow C(n) = \frac{n(n+1)(n+2)}{6} \in \Theta(\mathrm{n}^3)$$

Họ tên: Nguyễn Trung Dũng

MSSV: 19120486

Bài 1:

- Thao tác cơ sở: `a[pivot] ⇆ a[i]`
- Gọi $T(n)$ là số phép tính cần thực hiện với mảng n phần tử, ta có:

$$T(0) = 0, T(1) = 2, \dots$$

Mỗi vòng lặp thì thực hiện thao tác cơ sở 2 lần và thực hiện $T(n-1)$ 1 lần, lặp n lần

$$\Rightarrow T(n) = nT(n-1) + 2n$$

- Dùng hàm sinh, ta tìm được nghiệm của hệ thức đệ quy trên:

$$\begin{aligned} T(n) &= 2.n! \sum_{i=0}^{n-1} \frac{1}{i!} \\ &= 2.n! \left(\frac{1}{0!} + \frac{1}{1!} + \frac{1}{2!} + \cdots + \frac{1}{(n-1)!} \right) \\ &= 2\lfloor \mathrm{n!\, e} - 1 \rfloor \approx \Theta(\mathrm{n!}) \end{aligned}$$

Bài 2:

- Giải thuật tìm dạng thức liên kết:

```
// kiểm tra dạng fully connected mesh, tất cả các đỉnh phải có bậc = n-1
checkMesh(g[1..n][1..n]) {
    for (i = 1; i <= n; i++) {
        for (j = 1; j <= n; j++) {
            if (i != j && g[i][j] == false)
                return false;
        }
    }
    return true;
}
// kiểm tra dạng star, duy nhất 1 đỉnh có bậc n-1, tất cả các đỉnh còn lại bậc 1
checkStar(g[1..n][1..n]) {
    if (n == 1)
        return true;
    countHub = 0;
    for (i = 1; i <= n; i++) {
        deg = 0;                        // tính bậc của đỉnh i
        for (j = 1; j <= n; j++) {
            if (g[i][j] == true)
                deg++;
        }
        if (deg != 1 && deg != n-1)  // nếu tồn tại bậc khác 1 và khác n-1 thì
            return false;            // trái với điều kiện

        if (deg == n-1)              // đếm số lượng đỉnh có bậc n-1
            countHub++;
    }
    if (countHub != 1)               // nếu số lượng đỉnh bậc n-1 khác 1 thì trái
        return false;                // với điều kiện
    return true;
}
// kiểm tra dạng vòng, g chứa ít nhất 3 đỉnh, phải liên thông và tất cả các đỉnh
có bậc 2
checkRing(g[1..n][1..n]) {
    if (n < 3)
        return false;
    v[] = false for (i from 1 to n);
    duyet(g, v);                 // duyệt qua g 1 lần, có thể dùng bfs hoặc dfs

    for (i = 1; i <= n; i++)
        if (v[i] == false)       // nếu có đỉnh chưa duyệt thì g không liên thông
            return false;

    for (i = 1; i <= n; i++) {
        deg = 0;                 // tính bậc của đỉnh i
        for (j = 1; j <= n; j++) {
            if (g[i][j] == true)
                deg++;
        }
        if (deg != 2)            // nếu tồn tại đỉnh bậc khác 2 thì trái với điều
            return false;        // kiện
    }
    return true;
}
```

```
if (checkMesh(g))
    print(“G có dạng lưới đầy đủ”);
if (checkStar(g))
    print(“G có dạng sao”);
if (checkRing(g))
    print(“G có dạng vòng”);
```

- Các thao tác cơ sở đã được tô màu đỏ. Cả 3 thuật toán kiểm tra đều có độ phức tạp $\Theta(n^2)$

Bài 3:

- Giải thuật tìm 2 tập có tổng bằng nhau
- Ý tưởng: chọn bằng cách phát sinh dãy nhị phân, nếu một trạng thái đã thỏa điều kiện có tổng bằng một nửa tổng tất cả phần tử thì in ra và kết thúc chương trình.
- Mã giả:

```
partition(cur, a[1..n], list[]) {
    // nếu con trỏ cur nằm ngoài khoảng cần tính hoặc tổng các phần tử trong list
    lớn hơn sum(a)/2
    // trả về false
    if (cur > n+1 || sum(list) > sum(a)/2)
          return false;

    // nếu tổng trong list = tổng các số còn lại
    // in ra, trả về true và kết thúc chương trình
    if (sum(list) == sum(a)/2) {
        print(list);        // in các số trong list
        print(a \ list);    // in các số còn lại
        return true;
    }

    // nếu con trỏ cur chưa chạy hết mảng: 2 trường hợp:
    // chọn a[cur] hoặc không chọn a[cur] và tiếp tục xét các phần tử còn lại
    else if (cur <= n) {
        // chọn a[cur] và tiếp tục xét; khi xét các phần tử còn lại mà tìm thấy
        1 tập list phù hợp thì trả về true và kết thúc chương trình (không cần
        xét trường hợp không chọn a[cur])
        list.push(a[cur]);
        if (partition(cur + 1, a, list))
              return true;

        // loại phần tử a[cur] và tiếp tục xét
        list.pop();
        return partition(cur + 1, a, list);
    }

    // nếu cur = n+1 (kết thúc việc xét) nhưng sum(list) != sum(a)/2
    return false;
}
// khởi tạo list rỗng
list[] = none;
// kiểm tra tổng lẻ hoặc chia không được (nếu tổng chẵn và chia được thì chương
trình sẽ in ra 2 tập)
if (sum(a) mod 2 == 1 || !partition(1, a, list))
    print("Không có cách chia");
```

Bài 4:

a. Tổng của tất cả phần tử của ma trận:

$$S = 1 + 2 + 3 + \cdots + n^2$$

$$= \frac{n^2(n^2+1)}{2}$$

Mà mỗi dòng (cột) có tổng là t

$$\Rightarrow S = nt$$

$$\Rightarrow nt = \frac{n^2(n^2+1)}{2}$$

$$\Leftrightarrow \mathrm{t} = \frac{n(n^2+1)}{2}$$

b. Giải thuật phát sinh ma phương bậc n:

```
magicSquare(pivot, a[1..n][1..n]) {
    // nếu đã xét hết các phần tử trong một hoán vị thì kiểm tra nó có phải là
    một ma phương hay không
    // nếu phải thì in
    if (pivot == n^2)
        if (kiemTra(a)) // kiểm tra tổng các dòng/cột và 2 đường chéo bằng nhau
            print(a);

    // nếu chưa xét hết thì tiếp tục xét, lấy ý tưởng từ hàm Taohoanvi()
    // chuyển đổi pivot và i từ tọa độ logic sang tọa độ 2d
    else {
        pRow = (pivot - 1) div n + 1;
        pCol = (pivot - 1) mod n + 1;
        for(i = pivot; i <= n^2; i++){
            row = (i - 1) div n + 1;
            col = (i - 1) mod n + 1;

            a[pRow][pCol] ⇆ a[row][col];
            magicSquare(pivot + 1, a);
            a[pRow][pCol] ⇆ a[row][col];
        }
    }
}
// khởi tạo table a chứa nxn phần tử từ 1 tới n^2 theo thứ tự
a[][] = i for (i from 1 to n^2)
magicSquare(1, a)
```

Họ tên: Nguyễn Trung Dũng

MSSV: 19120486

Bài 1:

a)

- **Phép so sánh là thao tác cơ sở**

⇒ Hệ thức try hồi là:

$$C(n) = \begin{cases} C\left(\left\lfloor\frac{n}{2}\right\rfloor\right) + C\left(n - \left\lfloor\frac{n}{2}\right\rfloor\right) + 2, & \text{n} > 2 \\ 1 \quad , & \text{n} \leq 2 \end{cases}$$

Giả sử $n = 2^k$

$$\Rightarrow C(n) = \begin{cases} 2C\left(\frac{n}{2}\right) + 2, & \text{n} > 2 \\ 1 \quad , & \text{n} \leq 2 \end{cases}$$

⇒ Ta có:

$$\begin{aligned} C(n) &= 2C\left(\frac{n}{2}\right) + 2 \\ &= 2\left(2C\left(\frac{n}{4}\right) + 2\right) + 2 \\ &= \cdots \\ &= 2^{k-1}C(2) + 2 + 4 + 8 + \cdots + 2^{k-1} \\ &= 2^{k-1} + 2\frac{1 - 2^{k-1}}{1 - 2} \\ &= 2^k + 2^{k-1} - 2 \approx n \in \Theta(n) \end{aligned}$$

- **Phép so sánh `if(l >= r-1)` cũng là thao tác cơ sở**

⇒ Hệ thức try hồi là:

$$C(n) = \begin{cases} C\left(\left\lfloor\frac{n}{2}\right\rfloor\right) + C\left(n - \left\lfloor\frac{n}{2}\right\rfloor\right) + 3, & \text{n} > 2 \\ 2 \quad , & \text{n} \leq 2 \end{cases}$$

Giả sử $n = 2^k$, ta có:

$$C(n) = 2C\left(\frac{n}{2}\right) + 3$$

$$= 2\left(2C\left(\frac{n}{4}\right) + 3\right) + 3$$
$$= 2^{k-1}C(2) + 3 + 6 + 12 + \cdots + 3.2^{k-2}$$
$$= 2^{k-1} + 3\frac{1 - 2^{k-1}}{1 - 2}$$
$$= 2.2^k - 3 \approx n \in \Theta(n)$$

b) Tiếp cận có cùng độ phức tạp với chia để trị nhưng không dùng đệ quy:

```
MinMax(a[1..n]) {
    min = max = a[1];
    for (i = 2; i ≤ n; i++) {
        if (a[i] > max)
            max = a[i];
        if (a[i] < min)
            min = a[i];
    }
    return <min, max>
}
```

- Các thao tác cơ sở là các phép so sánh
- Có i chạy từ $2 \rightarrow n \Rightarrow$ tổng số lần thực hiện thao tác cơ sở: $2n - 2 \in \Theta(n)$

Bài 2:

a) Phép so sánh là thao tác cơ sở

- Trường hợp tốt nhất:

$$T(n) = \begin{cases} T\left(\left\lfloor\frac{n}{2}\right\rfloor\right) + T\left(\left\lceil\frac{n}{2}\right\rceil\right) + \left\lfloor\frac{n}{2}\right\rfloor, & n > 1 \\ 0, & n = 1 \end{cases}$$

Giả sử $n = 2^k$, ta có:

$$T(n) = 2T\left(\frac{n}{2}\right) + \frac{n}{2}$$
$$= 2\left(2T\left(\frac{n}{4}\right) + \frac{n}{4}\right) + \frac{n}{2}$$
$$= 2^k T(1) + \frac{n}{2} + \frac{n}{2} + \cdots + \frac{n}{2} \text{ (k lần)}$$
$$= k\frac{n}{2} = \frac{n}{2}.log_2 n \in \Theta(nlogn)$$

- Trường hợp xấu nhất:

$$T(n) = \begin{cases} T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + T\left(\left\lceil \frac{n}{2} \right\rceil\right) + (n-1), & \text{n} > 1 \\ 0 \quad , & \text{n} = 1 \end{cases}$$

Giả sử $n = 2^k$, ta có:

$$T(n) = 2T\left(\frac{n}{2}\right) + (n-1)$$

$$= 2\left(2T\left(\frac{n}{4}\right) + \left(\frac{n}{2} - 1\right)\right) + (n-1)$$

$$= 2^k T(1) + kn - (1 + 2 + 4 + \cdots + 2^{k-1})$$

$$= kn - \frac{1-2^k}{1-2} = nlog_2 n - n + 1 \in \Theta(nlogn)$$

b) Phép chuyển dời là thao tác cơ sở

⇒ Hệ thức truy hồi:

$$M(n) = \begin{cases} M\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + M\left(\left\lceil \frac{n}{2} \right\rceil\right) + n, & \text{n} > 1 \\ 0 \quad , & \text{n} = 1 \end{cases}$$

Giả sử $n = 2^k$, ta có:

$$M(n) = 2M\left(\frac{n}{2}\right) + n$$

$$= 2\left(2M\left(\frac{n}{4}\right) + \frac{n}{2}\right) + n$$

$$= 2^k M(1) + kn$$

$$= nlog_2 n \in \Theta(nlogn)$$

Bài 3:

a) Phân bố dữ liệu thỏa điều kiện không giảm dần

b) Trong mã nguồn của hàm mergeSort(), thêm vòng for chạy từ low+1 tới high kiểm tra xem dữ liệu đã thỏa điều kiện như vậy hay chưa

```
mergeSort(a[1 .. n], low, high) {
    if (low < high) {
        for (i = low+1; i <= high; i++) {
            if (a[i] < a[i-1])
                break;
            if (i == high)
                return;
        }
        mid = ⌊(low + high) / 2⌋;
        mergeSort(a, low, mid);
        mergeSort(a, mid + 1, high);
        merge(a, low, mid, high);
    }
}
```

Họ tên: Nguyễn Trung Dũng

MSSV: 19120486

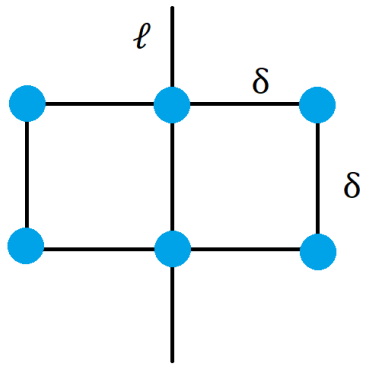
Bài 1:

- Ta có: $\delta = min\{\delta_L, \delta_R\}$ với δ_L, δ_R lần lượt là khoảng cách nhỏ nhất giữa 2 điểm bất kì ở bên trái và bên phải

  ⇒ Một cặp điểm bất kì ở cùng một phía của ℓ luôn có khoảng cách $d \geq \delta$

- Giả sử $\delta_L = \delta_R = \delta$

  ⇒ Cách phân bố sao cho có tối đa số điểm trong hình chữ nhật $2\delta \times \delta$ là các điểm đó phải nằm trên các điểm góc như hình sau:



  Vậy tồn tại tối đa 6 điểm trong vùng tìm kiếm diện tích $2\delta \times \delta$

Bài 2:

- Thao tác cơ sở:

```
tmpSum = moneyChange(coins, i) + moneyChange(coins, money - i);
```

  ⇒ Hệ thức try hồi:

$$T(n) = \begin{cases} \sum_{i=1}^{\lfloor n/2 \rfloor} \left(T(i) + T(n-i)\right) + \Theta\left(\frac{n}{2}\right) & n > 2 \\ 1 & n = 2 \\ 0 & n = 1 \end{cases}$$

$$\Rightarrow \mathrm{T(n)} = \begin{cases} \sum_{i=1}^{n-1} T(i) + \Theta\left(\frac{n}{2}\right) & n > 2 \\ 1 & n = 2 \\ 0 & n = 1 \end{cases}$$

- Có:

$$T(n) = \sum_{i=1}^{n-1} T(i) + \Theta\left(\frac{n}{2}\right) \quad (1)$$

$$T(n+1) = \sum_{i=1}^{n} T(i) + \Theta\left(\frac{n+1}{2}\right) \quad (2)$$

Lấy $(2) - (1)$:

$$T(n+1) - T(n) = T(n) + \Theta\left(\frac{n+1}{2}\right) - \Theta\left(\frac{n}{2}\right)$$

$$\Leftrightarrow T(n+1) = 2T(n) + \Theta\left(\frac{n+1}{2}\right) - \Theta\left(\frac{n}{2}\right)$$

$$\Rightarrow T(n) = 2T(n-1) + \Theta\left(\frac{n}{2}\right) - \Theta\left(\frac{n-1}{2}\right)$$

Có thể thấy rằng $\Theta\left(\frac{n}{2}\right)$ và $\Theta\left(\frac{n-1}{2}\right)$ có cùng bậc (bằng 1) và cùng hệ số trước n

$\Rightarrow \Theta\left(\frac{n}{2}\right) - \Theta\left(\frac{n-1}{2}\right) \approx \Theta(1)$

$$\Rightarrow T(n) = 2T(n-1) + \mathrm{c} \quad (\mathrm{c} \text{ là hằng số})$$
$$= 2^{n-2}T(2) + c(1 + 2 + 4 + \cdots + 2^{n-3})$$
$$= 2^{n-2} + c\frac{1 - 2^{n-2}}{1 - 2}$$
$$= (c+1)2^{n-2} - c$$
$$= \frac{c+1}{4}2^n - c \in \Theta(2^n)$$

Bài 3:

- Chương trình nhân 2 số nguyên dương n bit (C++):

```cpp
// tìm bit cao nhất (most significant bit)
int msb(int a) {
    if(a == 0)
        return 0;

    // giảm a cho tới khi chỉ còn lại bit cao nhất
    while(a - (a&-a))
        a -= a&-a;
```

```
    // tìm vị trí của bit cao nhất
    int i = 0;
    while((a & (1<<i)) == 0)
        i++;
    return i+1;
}

int Multiplication(int x, int y) {
    int n = max(msb(x), msb(y));   // tìm bit cao nhất
    if (x == 0 || y == 0)
        return 0;
    if (n <= 4)                    // nếu bit cao nhất <= 4 thì dùng built-in operator
        return x*y;

    n += n&1;                      // nếu n lẻ thì cho n += 1, nếu không thì n += 0
    int m = n/2;

    int xl = x >> m;
    int xr = x & ((1<<m) - 1);
    int yl = y >> m;
    int yr = y & ((1<<m) - 1);

    int ll = Multiplication(xl, yl);    // tính xl*yl
    int rr = Multiplication(xr, yr);    // tính xr*yr
    int lr = Multiplication(xl + xr, yl + yr) - (ll + rr);  // tính xl*yr + xr*yl

    int ans = (ll << n) + (lr << m) + rr;
    return ans;
}
```

Họ tên: Nguyễn Trung Dũng

MSSV: 19120486

Bài 1:

a) Chứng minh quy nạp: $J(2^k + i) = 2i + 1, i \in [0,2^k - 1]$

Có công thức truy hồi cho $J(n)$:

- $n = 1$:

$$J(n) = 1$$

- $n = 2h$:

$$J(2h) = 2J(h) - 1$$

- $n = 2h + 1$:

$$J(2h + 1) = 2J(h) + 1$$

- Xét $n = 1$, tức là $k = 0, i = 0$, ta thấy:

$$J(1) = 2.0 + 1 = 1 \quad \text{(đúng)}$$

- Xét n chẵn $(n = 2h)$:

- Chọn $n = 2^m + l, l \in [0,2^m - 2], l$ chẵn. Để ý: $\frac{l}{2} \in [0,2^{m-1} - 1]$

- Giả sử mệnh đề đúng với $n = h = 2^{m-1} + \frac{l}{2}$, tức là:

$$J(\mathrm{h}) = l + 1$$

- Cần chứng minh mệnh đề đúng với $n = 2h = 2^m + l$, tức là cần chứng minh:

$$J(2h) = 2l + 1$$

- Thật vậy:

$$J(2h) = 2J(h) - 1 = 2l + 1$$

- Xét n lẻ $(n = 2h + 1)$:

- Chọn $n = 2^m + l + 1, (l + 1) \in [0,2^m - 1], l$ chẵn.

- Giả sử mệnh đề đúng với $n = h = 2^{m-1} + \frac{l}{2}$, tức là:

$$J(\mathrm{h}) = l + 1$$

- Cần chứng minh mệnh đề đúng với $n = 2h + 1 = 2^m + l + 1$, tức là cần chứng minh:

$$J(2h + 1) = 2(l + 1) + 1 = 2l + 3$$

- Thật vậy:

$$J(2h + 1) = 2J(h) + 1 = 2l + 3$$

- Vậy:

$$J(2^k + i) = 2i + 1, i \in [0,2^k - 1]$$

b) Đặt $n = 2^k + i, i \in [0,2^k - 1]$.

Gọi b là dạng biểu diễn nhị phân của n, b' là phần còn lại của b sau khi bỏ đi bit cao nhất (có thể chứa các số 0 đứng đầu).

Vì $n = 2^k + i$ và $i < 2^k$ nên có thể nói:

- Bit cao nhất của b có vị trí k và
- b' cũng là dạng biểu diễn nhị phân của i.

Như vậy, b có dạng: $\overline{1b'}$ (bit cao nhất có vị trí là k)

$\Rightarrow \mathrm{rot\_left}(b) = \overline{b'1} = 2i + 1$ (phát biểu được chứng minh)

Bài 2:

- Gọi $f(n,k)$ là chi phí trung bình của hàm `Selection` với giá trị k nào đó
- Xác suất để 1 phần tử được chọn làm pivot: $\frac{1}{n}$

$$\Rightarrow f(n,k) = (n-1) + \frac{1}{n}\sum_{p=k+1}^{n} f(p-1,k) + \frac{1}{n}\sum_{p=1}^{k-1} f(n-p,k-p)$$

Khi đó, chi phí trung bình của giải thuật với k bất kì:

$$\begin{cases} f(n) = \frac{1}{n}\sum_{k=1}^{n} f(n,k) \quad (1) \\ f(1) = 0 \end{cases}$$

$* f(1) = 0$ vì mảng chỉ có 1 phần tử.

- Vậy ta có:

$$(1) \Leftrightarrow nf(n) = \sum_{k=1}^{n} f(n,k)$$

$$= n(n-1) + \frac{1}{n}\left(\sum_{k=1}^{n}\sum_{p=1}^{k-1} f(n-p,k-p) + \sum_{k=1}^{n}\sum_{p=k+1}^{n} f(p-1,k)\right)$$

- **<u>Xét tổng sau:</u>**

$$\sum_{k=1}^{n}\sum_{p=1}^{k-1} f(n-p,k-p)$$

$$= \sum_{k=1}^{n}\Big(f(n-1,k-1) + f(n-2,k-2) + \cdots + f\big(n-(k-1),k-(k-1)\big)\Big)$$

- Có:

$* k = 1$: $\emptyset$

$* k = 2$: $f(n-1,1)$

$* k = 3$: $f(n-1,2) + f(n-2,1)$

$* k = 4$: $f(n-1,3) + f(n-2,2) + f(n-3,1)$

$* \ldots$

$* k = n$: $f(n-1,n-1) + f(n-2,n-2) + \cdots + f(1,1)$

$$\Rightarrow \sum_{\boldsymbol{k=1}}^{\boldsymbol{n}}\sum_{\boldsymbol{p=1}}^{\boldsymbol{k-1}} \boldsymbol{f(n-p,k-p)} = \sum_{\boldsymbol{1\le j\le i\le n-1}} \boldsymbol{p(i,j)} = \sum_{\boldsymbol{p=1}}^{\boldsymbol{n-1}}\sum_{\boldsymbol{k=1}}^{\boldsymbol{p}} \boldsymbol{f(p,k)}$$

- **<u>Xét tổng sau:</u>**

$$\sum_{k=1}^{n}\sum_{p=k+1}^{n} f(p-1,k) = \sum_{k=1}^{n}\big(f(k,k) + f(k+1,k) + \cdots + f(n-1,k)\big)$$

- Có:

  $* k = 1$: $f(1,1) + f(2,1) + f(3,1) + \cdots + f(n-1,1)$

  $* k = 2$: $f(2,2) + f(3,2) + \cdots + f(n-1,2)$

  $* k = 3$: $f(3,3) + f(4,3) + \cdots + f(n-1,3)$

  $* \ldots$

  $* k = n-1$: $f(n-1, n-1)$

  $* k = n$: $\emptyset$

$$\Rightarrow \sum_{\boldsymbol{k=1}}^{\boldsymbol{n}} \sum_{\boldsymbol{p=k+1}}^{\boldsymbol{n}} \boldsymbol{f(p-1,k)} = \sum_{\boldsymbol{1 \le j \le i \le n-1}} \boldsymbol{p(i,j)} = \sum_{\boldsymbol{p=1}}^{\boldsymbol{n-1}} \sum_{\boldsymbol{k=1}}^{\boldsymbol{p}} \boldsymbol{f(p,k)}$$

- Vậy:

$$nf(n) = n(n-1) + \frac{2}{n} \sum_{p=1}^{n-1} \sum_{k=1}^{p} f(p,k) \quad (*)$$

- Mà:

$$f(n) = \frac{1}{n} \sum_{k=1}^{n} f(n,k)$$

$$\Rightarrow f(p) = \frac{1}{p} \sum_{k=1}^{p} f(p,k)$$

$$\Leftrightarrow \text{pf(p)} = \sum_{k=1}^{p} f(p,k) \quad (**)$$

$$(*), (**) \Rightarrow n^2 f(n) = n^2(n-1) + 2 \sum_{p=1}^{n-1} pf(p) \quad (2)$$

- Thế $n = n - 1$:

$$(n-1)^2 f(n-1) = (n-1)^2(n-2) + 2\sum_{p=1}^{n-2} pf(p) \quad (3)$$

Lấy $(2) - (3)$:

$n^2 f(n) - (n-1)^2 f(n-1) = 2(n-1)f(n-1) + (n-1)(3n-2)$

$\Leftrightarrow n^2 f(n) = (n-1)^2 f(n-1) + 2(n-1)f(n-1) + (n-1)(3n-2)$

$\Leftrightarrow f(n) = \frac{n^2-1}{n^2} f(n-1) + \frac{3n^2-5n+2}{n^2} < f(n-1) + 3$

$\Rightarrow f(n) < f(n-1) + 3 < f(n-2) + 3 + 3 < \cdots$

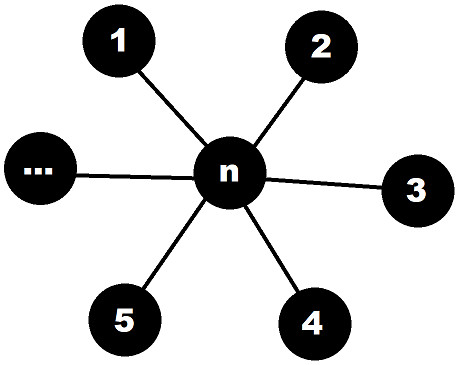
$< f(1) + 3(n-1) = 3(n-1)$

- **Kết luận:** chi phí trung bình của giải thuật: $f(n) \in \Theta(n)$

Bài 3:

- Giải thuật không thực hiện đúng nhiệm vụ (không thể kiểm tra tính liên thông của đồ thị vô hướng).
- Giải thích: đồ thị G với các đỉnh $\{1, \dots, n\}$ và ma trận kề $A[1..n][1..n]$ liên thông không có nghĩa là đồ thị $G' \subset G$ với các đỉnh $\{1, \dots, n-1\}$ và ma trận kề $A[1..n-1][1..n-1]$ liên thông.

Xét đồ thị sau:



Có thể thấy là đồ thị liên thông. Tuy nhiên, khi chỉ xét đến đỉnh thứ $n-1$ thì đồ thị không liên thông:

Dễ dàng thấy `Connected(A[1..2][1..2])` trả về `false`

⇒ Tương tự, `Connected(A[1..i][1..i])` trả về `false` $\forall i > 2$

⇒ `Connected(A[1..n][1..n]) = false` (sai)

- Vậy giải thuật không thực hiện đúng nhiệm vụ

Bài 4:

- Giải thuật tìm số vắng mặt:

```
find(arr[1..n], l, r) {
    if (l == r) {
        if (arr[l] == l)        // nếu số bị thiếu là n+1
            return l+1;
        return l;
    }
    m = ⌊(l+r)/2⌋;
    if (arr[m] == m)
        return find(arr, m+1, r);
    else
        return find(arr, l, m);
}
find(arr, 1, n);
```

- Các thao tác cơ sở đã được tô đỏ.

- Gọi $T(n)$ là số thao tác cần thực hiện với mảng n phần tử. Giả sử $n = 2^k$, ta có hệ thức hồi quy sau:

$$T(n) = \begin{cases} T\left(\frac{n}{2}\right) + 2 & n > 1 \\ 2 & n = 1 \end{cases}$$

- Thấy $a = 1, b = 2, d = 0$
- Theo định lí chủ, ta có: $T(n) \in \Theta(n^d \log n)$ hay $T(n) \in \Theta(\log n)$