Lab 02: Độ phức tạp thuật toán

Thời gian dự kiến: 02 tuần

1 Ước lượng độ phức tạp thuật toán

1.1 Giới thiệu

Ta đã biết, độ phức tạp (ở đây ta quan tâm độ phức tạp về mặt thời gian) của một thuật toán có thể được đánh giá qua hai khía cạnh *lý thuyết* và *thực nghiệm*. Với khía cạnh *thực nghiệm*, việc đánh giá được thực hiện bằng cách đo số lượng *phép toán cơ bản*, mà tùy vào bài toán ta lại có các *phép toán cơ bản* khác nhau (e.g. phép so sánh, phép gán, phép cộng, phép nhân, ..etc). Trong Lab 02, ta sẽ thực nghiệm đo độ phức tạp thời gian của một số thuật toán thường gặp, tập trung vào hai phép toán cơ bản là **phép gán** và **phép so sánh**.

Ví dụ: thuật toán tính tổng bình phương các số từ 1 đến n

- **Input:** một số nguyên dương – `n`
- **Output:** tổng các số $1^2 + 2^2 + .. + n^2$ – `squareSum` của đoạn $[1, n]$

Thuật toán được trình bày ở Thuật toán 1

```
// Calculate the sum of integer squares in [1, n]
int squareSum(int n) {
    int i = 1;
    int sum = 0;

    while (i <= n) {
        sum += i * i;
        i += 1;
    }

    return sum;
}
```

Mã nguồn 1: Thuật toán tính tổng bình phương các số từ 1 đến n

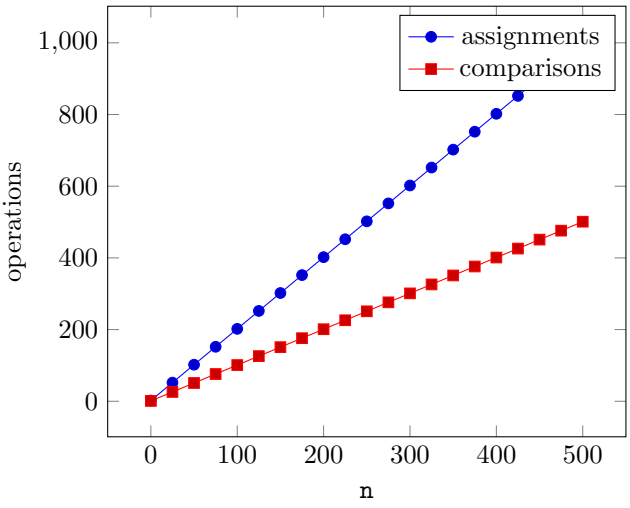
Theo lý thuyết, ta đã biết đoạn thuật toán này có độ phức tạp $\mathcal{O}(n)$ (*tuyến tính*), trong đó có đúng $2n + 2$ phép gán và $n + 1$ phép so sánh. Để kiểm chứng, ta dùng đoạn mã nguồn 2, với các biến đếm số phép gán `count_assignments` và đếm số phép so sánh `count_comparisons` được chèn vào các vị trí thích hợp sao cho **không làm thay đổi tính đúng đắn của thuật toán**:

```
// Calculate the sum of integer squares in [1, n]
int squareSum(int n, int& count_assignments, int& count_comparisons) {
    count_comparisons = 0;
    count_assignments = 0;

    int i = 1; count_assignments++;
    int sum = 0; count_assignments++;

    while (++count_comparisons && i <= n) {
        sum += i * i; ++count_assignments;
        i += 1; ++count_assignments;
    }

    return sum;
}
```

Mã nguồn 2: Thuật toán tính tổng bình phương các số từ 1 đến n, đã chèn các biến đếm số phép gán và phép so sánh

Trực quan hóa Sau khi đếm số phép gán và số phép so sánh của thuật toán `squareSum`, ta có thể trực quan hóa kết quả thông qua các bảng biểu và đồ thị như bên dưới:

n	assignments	comparisons
0	2	1
25	52	26
50	102	51
75	152	76
100	202	101
125	252	126
150	302	151
175	352	176
200	402	201
225	452	226
250	502	251
275	552	276
300	602	301
325	652	326
350	702	351
375	752	376
400	802	401
425	852	426
450	902	451
475	952	476
500	1,002	501



1.2 Thực hành

Yêu cầu Với mỗi bài tập, sinh viên được cung cấp một đoạn mã nguồn. Thực hiện các yêu cầu sau:

- Thêm các biến đếm số phép gán và số phép so sánh vào các vị trí thích hợp.
- Đếm số phép gán, số phép so sánh với `n` từ 0 đến 500.
- Ước lượng lý thuyết (Big-O, Big-Theta) cho số phép gán và số phép so sánh, sau đó dùng các số liệu thực nghiệm (dưới dạng bảng biểu, biểu đồ) để khẳng định ước lượng của mình là đúng.

Bài tập

(i) `sumHalf`

```
int sumHalf(int n) {
    int a = 0, i = n;
    while (i > 0) {
        a = a + i;
        i = i / 2;
    }
    return a;
}
```

Mã nguồn 3: `sumHalf`

(ii) `recursiveSquareSum`

```
int recursiveSquareSum(int n) {
    if (n < 1) return 0;
    return n * n + recursiveSquareSum(n - 1);
}
```

Mã nguồn 4: `recursiveSquareSum`

2 Thiết kế giải thuật

Một bài toán có thể có nhiều cách giải, trong đó một số cách giải lại *tối ưu* hơn cách giải khác. Trong phần này, sinh viên cần đề xuất hai thuật toán (trong đó một thuật toán phải *tối ưu* hơn thuật toán còn lại) để giải quyết một số bài toán cho trước, sau đó đánh giá các thuật toán này bằng các kĩ thuật đã nêu ở Phần 1.

Yêu cầu Với mỗi đề bài, sinh viên thực hiện các yêu cầu sau:

- Đưa ra **hai** giải thuật cho mỗi bài toán, trong đó đảm bảo có một giải thuật *tối ưu* hơn giải thuật còn lại. Sinh viên có thể đề xuất ra một thuật toán tối ưu, sau đó so sánh với một thuật toán kém tối ưu hơn.
- Đếm số phép gán, phép so sánh với các dữ liệu đầu vào khác nhau. **Lưu ý**: sự ổn định của dữ liệu đầu vào có thể ảnh hưởng đến kết quả của thuật toán, vì vậy sinh viên nên nêu rõ đặc điểm của dữ liệu đầu vào trong báo cáo.
- Ước lượng lý thuyết cho số phép gán và số phép so sánh, sau đó đưa ra các số liệu so sánh giữa hai thuật toán (Tương tự Phần 1). Nhận xét vì sao thuật toán tối ưu hơn thuật toán còn lại?

Các nhóm nhiều hơn 1 sinh viên **làm đủ** 2 bài có dấu (*). Các nhóm còn lại chọn 1 trong 2 bài.

Bài tập

(i) **Ước chung lớn nhất**: cho hai số nguyên u và v, tìm ước chung lớn nhất của u và v? Ước chung lớn nhất ở đây định nghĩa là số nguyên lớn nhất mà cả u và v cùng chia hết.

- **Input**: 2 số nguyên dương `u` và `v`, trong đó `u` > `v`
- **Output**: số nguyên `d` – ước chung lớn nhất của `u` và `v`
- **Hint**: Giải thuật Euclid

(ii) **Dãy con liên tiếp không giảm dài nhất**: cho mảng một chiều A gồm n phần tử, mỗi phần tử A_i là một số nguyên sao cho $-10^5 \leq A_i \leq 10^5$. Tìm một đoạn con $[a, b]$ sao cho $0 \leq a \leq b \leq n-1$ và $A_a \leq A_{a+1} \leq .. \leq A_{b-1} \leq A_b$.

- **Input**: số nguyên `n` và mảng `A` như đề bài
- **Output**: số nguyên `d` – độ dài dãy con không giảm dài nhất của `A`.
- **Hint**: bài này sinh viên có thể cài đặt thuật toán tối ưu (và đơn giản nhất) dùng duy nhất 1 vòng lặp, sau đó so sánh với thuật toán không tối ưu.

(iii) **Trung vị của mảng***: Cho một dãy số nguyên $x_1, x_2, .., x_n$ sao cho $-10^5 \leq x_i \leq 10^5$. Ta tiến hành chèn lần lượt các giá trị x_i vào một mảng A. Tại mỗi vị trí i (sau khi chèn x_i vào mảng A) cho biết giá trị *trung vị* của mảng A gồm i phần tử?

- **Input**: số nguyên `n` và n số nguyên `x1, x2, .., xn`.
- **Output**: mảng `B` có `n` phần tử, trong đó phần tử `B[i]` là trung vị của mảng `A[0..i]`. Lưu ý:
  - Nếu mảng có kích thước n là một số lẻ, phần tử *trung vị* của mảng là phần tử ở vị trí $\left\lceil \frac{n}{2} \right\rceil$ (phần tử ở vị trí chính giữa) sau khi sắp xếp lại các phần tử trong mảng theo thứ tự tăng dần.
  - Nếu mảng có chiều dài n là một số chẵn, phần tử *trung vị* của mảng là phần nguyên giá trị trung bình cộng của hai phần tử ở các vị trí lần lượt là $\frac{n}{2}$ và $\frac{n}{2} - 1$, sau khi sắp xếp lại mảng theo thứ tự tăng dần.

  Hint: thuật toán dễ cài đặt nhất sẽ phải sort mảng liên tục, vì vậy sinh viên cần lưu ý đếm các phép gán/so sánh trong thao tác sort. Với thuật toán tối ưu, một hướng xử lý là sử dụng 2 cấu trúc Heap. Sinh viên cân nhắc cài đặt một số thuật toán sort đơn giản/đã học.

(iv) **Số cặp nghịch thế***: Cho mảng A gồm n số nguyên sao cho $-10^5 \leq A_i \leq 10^5$. Một cặp *nghịch thế* của A là một cặp vị trí (i, j) sao cho $0 \leq i < j \leq n-1$ và $A_i > A_j$. Đếm tổng số cặp nghịch thế trong mảng A cho trước?

- **Input**: số nguyên `n` và mảng `A`
- **Output**: số nguyên `d` – tổng số cặp nghịch thế trong `A`.
- **Hint**: chia để trị.

A Một số lưu ý

A.1 Đánh giá lý thuyết

- Sinh viên có thể ước lượng sơ bộ các thuật toán thuộc $\mathcal{O}(f(n)), \Theta(g(n))$, không nhất thiết phải đưa ra công thức chính xác cho số phép gán/phép so sánh.
- Cẩn thận với các thuật toán dựa trên bộ dữ liệu đầu vào ngẫu nhiên (e.g. cho mảng A gồm n phần tử tùy thích). Với các trường hợp này, sinh viên chỉ cần đưa ra ước lượng sơ khởi cho trường hợp trung bình và dùng thực nghiệm để khẳng định ước lượng của mình là đúng.
- Các phân tích cho trường hợp tốt nhất, tệ nhất hoặc các chứng minh chặt chẽ cho độ phức tạp của thuật toán đều sẽ được điểm cộng. Lưu ý là một số kĩ thuật chứng minh sẽ được giới thiệu ở CSC14007 - Nhập môn Phân tích độ phức tạp thuật toán, nên phần này là tùy ý sinh viên.

A.2 Hướng dẫn nộp bài

Sinh viên tạo thư mục `MSSV1_MSSV2_MSSV3_MSSV4`, trong đó MSSV1, MSSV2, MSSV3 và MSSV4 lần lượt là mã số sinh viên của các thành viên trong nhóm. Thư mục này sẽ chứa

- File `report.pdf`: Báo cáo chi tiết, trong đó nêu rõ thông tin từng thực nghiệm (các kích thước `n` đã khảo sát; đặc điểm của dữ liệu – sinh ngẫu nhiên / tự cho / có thứ tự ..etc; các bảng biểu, biểu đồ minh họa kết quả thực nghiệm). Các mã nguồn, hình ảnh, bảng biểu từ internet hoặc từ một tài liệu nào khác phải có dẫn nguồn cụ thể.
- Thư mục `code`: Mã nguồn cài đặt các chương trình thực nghiệm. Chú ý đặt tên các chương trình thực nghiệm sao cho giáo viên dễ nhận biết.
- File `readme.txt`: Hướng dẫn chạy các chương trình thực nghiệm.

Khi nộp, sinh viên nén thư mục thành tập tin `MSSV1_MSSV2_MSSV3_MSSV4.zip` rồi submit lên Moodle.