BÁO CÁO THỰC HÀNH MÔN PHÂN TÍCH THUẬT TOÁN

Lab 2 - Tuần 6

Đinh Anh Huy - 18110103

Bài toán 1. Cho một số tự nhiên x và A là 1 mảng N số tự nhiên đôi một khác nhau. Hãy thiết kế một thuật toán có độ phức tạp $O(N \log N)$ theo thời gian để kiểm tra xem có tồn tại (i, j) sao cho $A[i] + A[j] = x$.

1. Input: N, A, x.
2. Output: (i, j).

Trong đó cho $x = 50$.

- N tăng dần theo thứ tự $10, 29, 30, ..., 1000$.
- A được tạo ngẫu nhiên sao cho $A[i] \in [1, 2, ..., 10000], A[i] \neq A[j], \forall i, j \in \{1, ..., 1000\}$.
- Chứng minh rằng thuật toán đưa ra thoả yêu cầu đề bài.
- Với mỗi $N \in \{10, 20, ..., 1000\}$, chọn $x = 50$ tạo ngẫu nhiên A và tính thời gian trung bình để kiểm tra $\exists (i, j) : A[i] + A[j] = x$. Gọi giá trị đó là $T(N)$. So sánh kết quả thực nghiệm và kết quả lý thuyết: $T(N) = O(N \log N)$

Lời giải

Thuật toán kiểm tra sự tồn tại của cặp giá trị (i, j) sao cho $A[i] + A[j] = x$:

Algorithm 1: Thuật toán kiểm tra sự tồn tại của $(i, j) : A[i] + A[j] = x$.

Function `Check-Existence`(A, N, x)**:**

```
A ← mergeSort(A);
for i = 0 to N − 1 do
    foundIndex ← binarySearch(A[i + 1, N − 1], x − A[i]);
    if foundIndex = −1 then
        continue;
    end
    return (i, foundIndex);
end
return −1;
```

Ý tưởng thuật toán:

1. Sắp xếp mảng theo thứ tự tăng dần bằng thuật toán *Merge Sort*.

2. Ứng với từng phần tử trong mảng A, ta dùng thuật toán *Binary Search* để tìm giá trị thoả mãn tổng giá trị cần tìm và giá trị đang xét bằng x. Như vậy ta chỉ cần tìm giá trị $x - A[i]$ trong phần còn lại của mảng tính từ phần tử đang xét, với $A[i]$ là phần tử đang xét trong mảng A.

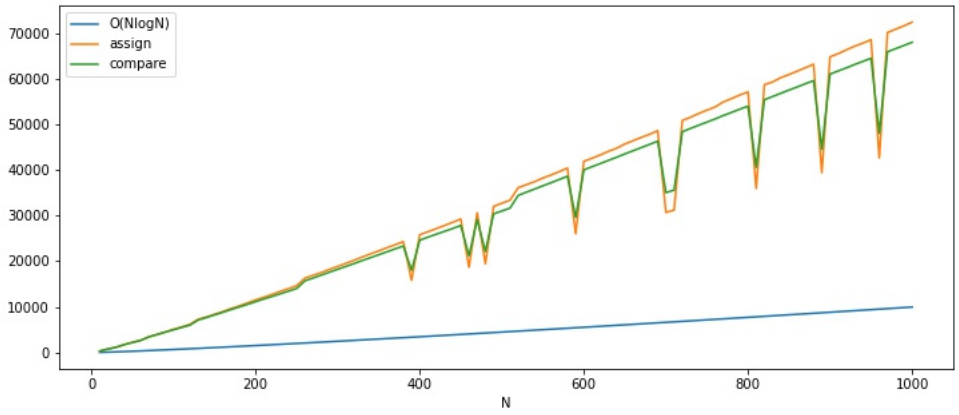
3. Trả về cặp giá trị (i, j) đầu tiên nếu tồn tại, ngược lại lặp lại bước 2 cho đến hết mảng. Nếu vẫn không tìm được cặp giá trị thoả đề bài trả về -1.

Độ phức tạp của thuật toán:

Gọi $T(N)$ là độ phức tạp của thuật toán trên. Do độ phức tạp của thuật toán *Merge Sort* là $O(N \log N)$ và của thuật toán *Binary Search* là $O(\log N)$ nên ta có

$$\begin{aligned} T(N) &= O(N \log N) + \sum_{i=0}^{N-1} O(\log(N - 1 - i - 1 + 1)) \\ &= O(N \log N) + \sum_{i=0}^{N-1} O(\log(N - i - 1)) \\ &\leq O(N \log N) + \sum_{i=0}^{N-1} O(\log(N)) \\ &= O(N \log N) + N \times O(\log N) = 2 \times O(N \log N) = O(N \log N). \end{aligned}$$

Ta có biểu đồ biểu diễn số phép so sánh và số phép gán của thuật toán trên theo N so với đường $O(N \log N)$ như sau



Từ biểu đồ trên, ta có thể thấy rằng đường biểu diễn số phép gán và số phép so sánh từ thuật toán trên tuy không cùng phương với đường $O(N \log N)$ nhưng cũng có dạng tượng tự như đường $O(N \log N)$ và cùng xuất phát từ gốc toạ độ. Sự chênh lệch khiến cho các đường không cùng phương với nhau có thể là do quá trình xây dựng thuật toán dẫn đến số lượng phép gán và phép so sánh biến thiên khá lớn so với đường $O(N \log N)$.