AN ALGORITHM HELPS REDUCE THE NUMBER OF COMPARISONS FOR SORTING X + Y

Pham Van Viet
Le Quy Don Technical University

ARTICLE INFO

Received: 17/7/2024
Revised: 30/9/2024
Published: 30/9/2024

KEYWORDS

Data structures and algorithms
Sorting algorithms
Heap sort
Sorting X + Y
Algorithm analysis

ABSTRACT

This paper proposes an efficient algorithm for comparison reduction for sorting X + Y. The proposed algorithm first sorts set X and set Y individually, then repeatedly selects a pair of elements from X and Y and adds the pair into X + Y in ascending order of sums of pairs. The algorithm is designed with a technique able to limit the number of pairs to be considered for each selection. The technique is able to limit the number of elements of X to be considered and considers only one element of Y to be paired with each element of X to select the pair with the smallest sum among the pairs that haven't been selected. A heap data structure used to store and update pairs to be considered helps operations in selecting a pair only need to run in O(log(n)) time, and the total running time of the algorithm is $O(n^2 log(n))$. The experimental results show that the number of comparisons and the running time of the proposed algorithm are significantly less than those of a traditional heapsort based approach.

MỘT THUẬT TOÁN GIÚP GIẢM THIỂU SỐ PHÉP SO SÁNH CHO BÀI TOÁN SẮP XẾP X + Y

Phạm Văn Việt
Trường Đại học Kỹ thuật Lê Quý Đôn

THÔNG TIN BÀI BÁO

Ngày nhận bài: 17/7/2024
Ngày hoàn thiện: 30/9/2024
Ngày đăng: 30/9/2024

TỪ KHÓA

Các cấu trúc dữ liệu và thuật toán
Các thuật toán sắp xếp
Thuật toán sắp xếp vun đống
Sắp xếp X + Y
Phân tích thuật toán

TÓM TẮT

Bài báo này đề xuất một thuật toán hiệu quả để giảm thiểu số lượng phép so sánh cho bài toán sắp xếp X + Y. Thuật toán đề xuất trước hết sắp xếp riêng các tập X và Y, sau đó tiến hành chọn từng cặp phần tử từ tập X và tập Y và thêm vào tập X + Y theo thứ tự tổng tăng dần của các cặp. Thuật toán được thiết kế với kỹ thuật có khả năng hạn chế số cặp phải xét trong mỗi lần chọn. Kỹ thuật có khả năng hạn chế số phần tử thuộc X được xét và chỉ xét duy nhất một phần tử thuộc Y để ghép cặp với mỗi phần tử thuộc X để chọn ra cặp có tổng nhỏ nhất trong các cặp chưa được chọn. Cấu trúc đống được sử dụng để lưu trữ và cập nhật lại các cặp được xét giúp các thao tác trong lựa chọn một cặp phần tử chỉ cần chạy trong thời gian O(log(n)) và tổng thời gian chạy của thuật toán là $O(n^2 log(n))$. Kết quả thực nghiệm cho thấy số lượng phép so sánh và thời gian chạy của thuật toán đề xuất nhỏ hơn đáng kể so với các giá trị số này của cách tiếp cận dựa trên thuật toán sắp xếp vun đống truyền thống.

DOI: https://doi.org/10.34238/tnu-jst.10781

Email: v.v.pham2012@gmail.com

1. Giới thiệu

Bài báo này đề xuất thuật toán giải quyết bài toán sắp xếp X + Y. Bài toán có đầu vào là 2 tập X và Y có số phần tử như nhau và yêu cầu đặt ra là đưa ra một tập tất cả các cặp phần tử gồm một phần tử thuộc X và một phần tử thuộc Y; các cặp này được sắp xếp theo thứ tự tổng của mỗi cặp tăng dần [1]. Ví dụ ta có hai tập đầu vào là X = {0, 3, 4} và Y = {1, 5, 8} thì đầu ra là dãy các cặp được sắp xếp theo thứ tự tổng tăng dần của mỗi cặp như sau:

{0, 1}, {3, 1}, {0, 5}, {4, 1}, {0, 8}, {3, 5}, {4, 5}, {3, 8}, {4, 8}.

Đây là bài toán sắp xếp có nhiều ứng dụng. Bài toán này có thể ứng dụng trong tìm kiếm đường bay giữa hai điểm, có đi qua một điểm trung gian sao cho tổng chi phí là nhỏ nhất [2]. Nhân hai đa thức của một biến duy nhất cũng có thể sử dụng sắp xếp X + Y [3]. Sắp xếp X + Y có thể ứng dụng trong thiết kế VLSI [4]. VLSI là quy trình tạo mạng điện tích hợp (IC) bằng cách kết hợp hàng triệu hay hàng tỷ bóng bán dẫn vào một chíp. Trong một thuật toán trong thiết kế VLSI, sắp xếp X + Y là một chương trình con tốn kém nhất. Sắp xếp X + Y cũng được áp dụng để giải bài toán 3SUM [5] hay ứng dụng trong lĩnh vực xử lý ảnh [6], [7]. Một số ứng dụng không nhất thiết phải sắp xếp tất cả các phần tử trong X + Y, mà chỉ cần chọn ra k phần tử có tổng các phần tử nhỏ nhất, do đó ta còn có bài toán gần gũi với sắp xếp X + Y là lựa chọn k phần tử có tổng các phần tử nhỏ nhất trong tập $X_1 + X_2 + ... + X_m$ [8], [9].

Các tác giả trong [10] đã tóm tắt các nghiên cứu tiêu biểu giải quyết bài toán sắp xếp X + Y. Với tập X và Y có kích thước bằng n, X + Y có thể sắp xếp bằng thuật toán truyền thống như thuật toán sắp xếp vun đống... với độ phức tạp là $n^2\log(n)$. Đề xuất của Jean-Luc Lambert [11] nói rằng có thể sắp xếp X + Y với $O(n^2)$ phép so sánh, nhưng lại dựa vào khả năng sắp xếp R - R từ tập R + R đã sắp xếp mà không cần phép so sánh nào. Việc sắp xếp R - R có thể thực hiện bằng thuật toán truyền thống, trong đó việc so sánh r - r' và s - s' không thực hiện mà thông qua việc xác định thứ tự của r + s' và s + r' trong R + R. Việc xác định thứ tự thực hiện thế nào nếu không cần so sánh không được trình bày. Nhưng tác giả kết luận thủ tục sắp xếp R - R vẫn có độ phức tạp là $n^2\log(n)$. Kane [12] đã chứng tỏ rằng X + Y có thể sắp xếp sử dụng $O(n\log^2 n)$ phép so sánh, nhưng cách cài đặt thuật toán một cách hiệu quả cũng không được mô tả. Trong thực tế không có cách tiếp cận được biết nào nhanh hơn cách tiếp cận dựa trên thuật toán sắp xếp truyền thống với độ phức tạp là $O(n^2\log(n^2)) = O(n^2\log(n))$ [8].

Bài báo này đề xuất thuật toán để sắp xếp X + Y với mục tiêu giảm thiểu số phép so sánh. Kết quả thực nghiệm cho thấy thuật toán đề xuất có thể giảm được đáng kể số phép so sánh, so với cách tiếp cận dựa trên thuật toán sắp xếp vun đống truyền thống.

Nội dung của các phần tiếp theo như sau: Phần 2 trình bày cách tiếp cận dựa trên thuật toán sắp xếp vun đống và thuật toán đề xuất để giải bài toán X + Y. Phần 3 là thực nghiệm và đánh giá. Phần cuối cùng đưa ra các kết luận.

2. Các thuật toán

Trong bài báo này, chúng tôi đề xuất một thuật toán mới để sắp xếp X + Y và so sánh với cách tiếp cận dựa trên thuật toán sắp xếp vun đống truyền thống bằng thực nghiệm. Cả hai cách tiếp cận sẽ được trình bày trong phần này.

2.1. Thuật toán sắp xếp vun đống và áp dụng sắp xếp X + Y

Phần này trình bày thuật toán sắp xếp vun đống và cách sử dụng để sắp xếp X + Y.

2.1.1. Thuật toán sắp xếp vun đống

Thuật toán sắp xếp vun đống [13] có số phép so sánh trong trường hợp xấu nhất là O(nlog(n)) là một thuật toán mạnh trong nhóm các thuật toán truyền thống như sắp xếp nhanh hay hòa nhập với số phép so sánh trong trường hợp xấu nhất lần lượt là $O(n^2)$ và O(nlog(n)). Thuật toán sắp xếp vun đống cho một danh sách các phần tử thực hiện hai công việc. Trước hết thuật toán tạo

đống trên cùng danh sách các phần tử ban đầu. Mối quan hệ giữa các phần tử trong đống được xác định qua chỉ số của nó. Trong một danh sách các phần tử được đánh số từ 0, phần tử thứ i có cha ở vị trí [(i – 1)/2], con trái ở vị trí 2 × i + 1 và con phải ở vị trí 2 × i + 2 nếu có. Giá trị của phần tử cha lớn hơn hoặc bằng giá trị của phần tử con. Phần tử ở vị trí đầu tiên của danh sách không là con của phần tử nào và có giá trị lớn nhất. Tiếp đó, thuật toán phân danh sách thành hai phần, phần cuối danh sách gồm các phần tử đã được sắp xếp theo thứ tự tăng dần và phần đầu danh sách gồm các phần tử chưa được sắp xếp, nhưng vẫn duy trì cấu trúc đống. Thuật toán lần lượt đổi chỗ phần tử ở vị trí đầu tiên và cuối cùng của phần đầu danh sách; sau đó phần tử cuối cùng của phần đầu được kết nạp vào phần sau. Phần đầu khi đó có thể không đảm bảo được trật tự đống do phần tử trên đỉnh của đống (ở vị trí đầu tiên của phần đầu) nhỏ hơn các phần tử con của nó. Thuật toán sẽ đổi chỗ phần tử đầu tiên với phần tử con lớn hơn của nó cho đến khi đạt được trật tự của đống.

2.1.2. Áp dụng sắp xếp X + Y

Bài báo áp dụng thuật toán sắp xếp vun đống để sắp xếp X + Y như sau. Đầu tiên thuật toán đưa tất cả n^2 bộ (x, y) vào một danh sách, trong đó x là phần tử thuộc X và y là phần tử thuộc Y. Sau đó thực hiện thuật toán sắp xếp vun đống trên danh sách các bộ (x, y) theo tổng của x và y. Thuật toán có số phép so sánh là $O(n^2\log(n^2)) = O(n^2\log(n))$.

2.2. Thuật toán đề xuất

Phần này trình bày thuật toán đề xuất và ví dụ minh họa cho các bước chạy của thuật toán.

2.2.1 Thuật toán

Thuật toán 1 là thuật toán đề xuất thể hiện bằng mã giả Python. Thuật toán có đầu vào là hai tập X và Y có n phần tử, được lưu trong hai danh sách và được đánh số từ 0. Thuật toán cần tạo ra tập X + Y được lưu trong danh sách XY có n × n phần tử, mỗi phần tử gồm một phần tử thuộc X và một phần tử thuộc Y. Các phần tử trong danh sách XY được sắp xếp theo thứ tự tổng tăng dần của hai phần tử trong mỗi phần tử. Thuật toán trước hết sắp xếp riêng danh sách X và Y sử dụng thuật toán sắp xếp vun đống. Thuật toán lưu trữ và cập nhật lại các cặp phần tử phải xét để chọn ra cặp có tổng nhỏ nhất trong các cặp chưa được đưa vào XY trong danh sách XYS. Mỗi phần tử thuộc XYS gồm 3 thuộc tính: xID là chỉ số một phần tử thuộc X, yID là chỉ số một phần tử thuộc Y và s là tổng của hai phần tử. Tuy nhiên, để đơn giản trong trình bày, thuật ngữ cặp chỉ số của hai phần tử hoặc cặp phần tử được dùng thay thế cho bộ 3 thành phần trong một số nội dung dưới đây, vì tổng của hai phần tử thuộc X và Y có thể tính được từ cặp chỉ số. Danh sách XYS có cấu trúc đống nhỏ nhất theo giá trị tổng s của hai phần tử. Danh sách XYS ban đầu chỉ có một phần tử là (0, 0, X[0] + Y[0]) là phần tử có tổng s nhỏ nhất trong các phần tử. Thuật toán cũng lưu chỉ số lớn nhất của phần tử thuộc X trong XYS trong biến lastXID, ban đầu là 0. Tiếp đó, thuật toán thực hiện vòng lặp for (dòng 7) với biến chạy step trong n × n lần. Mỗi bước lặp chọn ra cặp có tổng nhỏ nhất trong các cặp chưa được chọn để cho vào XY. Thuật toán đưa ra kỹ thuật hạn chế số cặp phần tử phải xét trong mỗi lần chọn. Mỗi phần tử x thuộc X chỉ được xét ghép cặp với đúng một phần tử thuộc Y, là phần tử nhỏ nhất chưa được ghép cặp với x. Mỗi lần cặp chỉ số xID, yID ở gốc của đống XYS được chọn: nếu yID < n - 1, thì cặp này được thay thế bằng cặp xID, yID + 1 và được đưa xuống cho tới khi đạt được trật tự đống (các cặp ghép xID với chỉ số phần tử thuộc Y nhỏ hơn yID không được xét vì cặp phần tử tương ứng đã được đưa vào XY, các cặp với chỉ số phần tử thuộc Y lớn hơn yID + 1 cũng không được xét vì tổng các phần tử tương ứng lớn hơn hoặc bằng X[xID] + Y[yID + 1]); nếu yID = n - 1 thì cặp (xID, yID) bị loại khỏi đống, xID sẽ không còn xuất hiện lại trong XYS vì yID = n - 1 đồng nghĩa với tất cả các cặp có xID đã được chọn. Ngoài ra nếu xID là chỉ số lớn nhất trong các chỉ số phần tử thuộc X trong XYS (xID = lastXID) và lastXID < n – 1 thì tăng lastXID lên một đơn vị và bổ sung thêm cặp chỉ số lastXID, 0 vào XYS. Danh sách các cặp phần tử được xét XYS không có cặp nào

có xID > lastXID do tổng các phần tử tương ứng lớn hơn hoặc bằng X[lastXID] + Y[0], tức là XYS chỉ chứa các xID ≤ lastXID. Các kỹ thuật này giúp hạn chế số phần tử thuộc X được xét trong mỗi lần chọn.

Ngoài ra, cách lưu trữ các cặp phần tử được xét trong cấu trúc đống giúp việc chọn ra cặp nhỏ nhất được thực hiện trong thời gian O(1). Các thao tác thay thế cặp phần tử ở gốc, xóa cặp phần tử ở gốc và thêm một cặp phần tử, tương đương với thời gian đưa một cặp phần tử xuống phía dưới hay lên phía trên của đống cho đến khi đạt được trật tự của đống là O(log(n)) [13]. Bằng những kỹ thuật như vậy thuật toán đã giúp giảm số phép so sánh và thời gian tính toán để giải bài toán sắp xếp X + Y.

Thuật toán 1. *Thuật toán sắp xếp*

Đầu vào: Hai danh sách đầu vào X và Y. Mỗi danh sách có n phần tử với chỉ số bắt đầu từ 0.
Đầu ra: Một danh sách đầu ra XY được sắp xếp có n × n phần tử. Mỗi phần tử chứa một phần tử thuộc X và một phần tử thuộc Y.

```
1  Khởi tạo danh sách XY rỗng
2  Sắp xếp X bằng thuật toán sắp xếp vun đống
3  Sắp xếp Y bằng thuật toán sắp xếp vun đống
4  Khởi tạo đống XYS lưu các bộ (xID, yID, X[xID] + Y[yID]) được xét để chọn ra cặp phần tử có tổng nhỏ nhất
5  Thêm bộ (0, 0, X[0] + Y[0]) là bộ có tổng các phần tử nhỏ nhất vào XYS
6  Đặt lastXID = 0  # lastXID lưu chỉ số lớn nhất của phần tử thuộc X trong XYS
7  for step in range(0, n*n)
8      Lấy phần tử XYS[0] ở gốc của đống XYS và đưa cặp phần tử (x, y) tương ứng với cặp chỉ số thuộc XYS[0] vào XY
9      Lưu XYS[0].xID và XYS[0].yID vào biến xID và yID
10     if (yID < n – 1):
11         Thay thế phần tử XYS[0] bằng phần tử (xID, yID + 1, X[xID] + Y[yID + 1])
12         Đưa (xID, yID + 1, X[xID] + Y[yID + 1]) xuống dưới cho đến khi đạt được trật tự đống nhỏ nhất
13     else:  # trường hợp yID == n - 1
14         Xóa XYS[0] ở gốc khỏi đống XYS
15     if (xID == lastXID and lastXID < n - 1):
16         lastXID += 1
17         Thêm phần tử (lastXID, 0, X[lastXID] + Y[0]) vào đống XYS
```

2.2.2. Ví dụ minh họa

Sau dòng lệnh 6, ta có danh sách XY rỗng, danh sách X và Y đã được sắp xếp, đống XYS có một phần tử là (0, 0, X[0] + Y[0]), chỉ số lớn nhất của phần tử thuộc X trong XYS lưu trong biến lastXID bằng 0. Để minh họa cho các bước của vòng lặp for (dòng 7) với biến chạy step của thuật toán đề xuất, chúng tôi sử dụng đầu vào X = {0, 3, 4} và Y = {1, 5, 8} đã đề cập trong phần giới thiệu. Danh sách X và Y trong trường hợp này có kích thước n = 3 và đã được sắp xếp.

Bảng 1. *Bảng minh họa các bước chạy thuật toán đề xuất*

step	lastXID	XYS	xID, yID, s	x, y, s
0	0	(0, 0, 1)	0, 0, 1	0, 1, 1
1	1	(1, 0, 4)(0, 1, 5)	1, 0, 4	3, 1, 4
2	2	(0, 1, 5)(1, 1, 8)(2, 0, 5)	0, 1, 5	0, 5, 5
3	2	(2, 0, 5)(1, 1, 8)(0, 2, 8)	2, 0, 5	4, 1, 5
4	2	(1, 1, 8)(2, 1, 9)(0, 2, 8)	1, 1, 8	3, 5, 8
5	2	(0, 2, 8)(2, 1, 9)(1, 2, 11)	0, 2, 8	0, 8, 8
6	2	(2, 1, 9)(1, 2, 11)	2, 1, 9	4, 5, 9
7	2	(1, 2, 11)(2, 2, 12)	1, 2, 11	3, 8, 11
8	2	(2, 2, 12)	2, 2, 12	4, 8, 12

Bảng 1 thể hiện giá trị của các biến ở đầu mỗi bước lặp, trong đó: cột "step" là giá trị của biến step (chính là chỉ số bước lặp), cột "lastXID" là chỉ số lớn nhất của phần tử X trong XYS, cột "XYS" là đống lưu các bộ (xID, yID, X[xID] + Y[yID]) được xét để chọn cặp có tổng nhỏ nhất trong các cặp chưa được chọn để đưa vào danh sách được sắp xếp XY, cột "xID, yID, s" thể hiện bộ chỉ số hai phần tử thuộc X và Y được chọn để đưa vào XY và tổng hai phần tử, cột "x, y, s" thể hiện bộ giá trị của hai phần tử thuộc X và Y được chọn để đưa vào XY và tổng hai phần tử.

Vòng lặp for kết thúc sau 3 × 3 = 9 bước lặp, ứng với các giá trị step từ 0 đến 8. Ta có các cặp đã lần lượt được chọn để thêm vào XY là {0, 1}, {3, 1}, {0, 5}, {4, 1}, {3, 5}, {0, 8}, {4, 5}, {3, 8}, {4, 8}. Các cặp này đã được sắp xếp theo thứ tự tổng tăng dần.

Bảng 2. *So sánh thuật toán đề xuất và cách tiếp cận truyền thống*

n	Đề xuất		Truyền thống		Đề xuất/ Truyền thống	
	#So sánh	Thời gian	#So sánh	Thời gian	#So sánh	Thời gian
100	94.778	31	235.047	69	40,32%	44,93%
200	449.945	126	1.099.708	304	40,91%	41,45%
300	1.106.137	294	2.687.296	887	41,16%	33,15%
400	2.095.983	562	5.039.003	1.664	41,60%	33,77%
500	3.419.819	961	8.187.691	2.868	41,77%	33,51%
600	5.095.811	1.500	12.185.630	4.371	41,82%	34,32%
700	7.145.070	1.996	17.001.861	6.398	42,03%	31,20%
800	9.567.964	2.659	22.714.673	8.649	42,12%	30,74%
900	12.362.638	3.489	29.310.098	11.173	42,18%	31,23%
1000	15.530.542	4.426	36.747.976	14.480	42,26%	30,57%

3. Thực nghiệm và đánh giá

Bài báo tiến hành thực nghiệm với thuật toán "Đề xuất" và cách tiếp cận dựa trên thuật toán sắp xếp vun đống "Truyền thống" đã mô tả ở Phần 2. Mỗi thuật toán được chạy với các tập đầu vào X và Y có các kích thước khác nhau n = 100, 200,..., 1000 sử dụng máy tính xách tay có bộ xử lý Intel(R) Core(TM) i3-1005G1 CPU @ 1.20GHz 1.19 GHz và RAM 4 GB. Với mỗi kích thước n, mỗi thuật toán chạy 20 lần với 20 bộ dữ liệu ngẫu nhiên của các tập X và Y. Giá trị của các phần tử trong tập X và Y nằm trong khoảng từ 0 đến 50000. Với mỗi kích thước n, số phép so sánh trung bình và thời gian chạy trung bình của 20 lần chạy được ghi nhận lại cho thuật toán "Đề xuất" và "Truyền thống". Các phép so sánh ở đây bao gồm so sánh giữa các phần tử thuộc X với nhau, giữa các phần tử thuộc Y với nhau và giữa tổng các cặp x + y với nhau. Kết quả thực nghiệm thể hiện trong Bảng 2, trong đó cột "#So sánh" và "Thời gian" là số phép so sánh trung bình và thời gian trung bình tính bằng ms (mili giây) của 20 lần chạy, cột "Đề xuất/Truyền thống" là tỷ lệ phần trăm số phép so sánh trung bình và thời gian chạy trung bình của thuật toán đề xuất trên các giá trị số đó của cách tiếp cận dựa trên thuật toán sắp xếp vun đống truyền thống.

Bảng 2 cho thấy số phép so sánh trung bình và thời gian chạy trung bình của thuật toán đề xuất ít hơn đáng kể so với các giá trị số đó của cách tiếp cận truyền thống, chỉ bằng chưa đến 43% và chưa đến 45% các giá trị số đó của cách tiếp cận truyền thống. Tính trung bình thì số phép so sánh trung bình và thời gian chạy trung bình của thuật toán đề xuất chỉ bằng khoảng 42% và 34% các giá trị số đó của các tiếp cận truyền thống. Kết quả này có được là do thuật toán đề xuất đã được thiết kế với kỹ thuật hạn chế các cặp được xét trong mỗi lần chọn ra cặp (x, y) có tổng bé nhất trong các cặp còn lại để đưa vào tập X + Y. Trong mỗi lần chọn, kỹ thuật có khả năng hạn chế số phần tử thuộc X được xét và chỉ xét duy nhất một phần tử thuộc Y để ghép cặp với mỗi phần tử thuộc X. Bên cạnh đó, bằng cách lưu trữ các cặp phần tử được xét trong danh sách tổ chức dưới dạng cấu trúc đống nhỏ nhất theo tổng của cặp phần tử, việc chọn cặp phần tử có tổng nhỏ nhất thực hiện trong thời gian O(1) vì cặp này luôn nằm ở gốc của đống, cũng chính là ở đầu danh sách. Việc thay thế phần tử và xóa phần tử sau khi được chọn, và việc thêm phần tử được xét vào đống cũng thực hiện với thời gian O(log(n)). Như vậy mỗi bước lặp để chọn ra một

cặp phần tử và cập nhật lại đống các cặp phần tử được xét cho bước lặp tiếp theo chỉ thực hiện với thời gian O(log(n)). Thuật toán thực hiện n × n bước lặp với thời gian là $O(n^2log(n))$. Trong các hoạt động từ dòng 1 đến dòng 6 của Thuật toán 1, hoạt động ở dòng 2 và 3 thực hiện sắp xếp riêng tập X và Y bằng thuật toán sắp xếp vun đống có độ phức tạp cao nhất là O(nlog(n)). Thế nên, độ phức tạp chung của cả Thuật toán 1 là $O(n^2log(n))$.

4. Kết luận

Bài báo đã đề xuất một thuật toán hiệu quả để sắp xếp X + Y để giảm thiểu số phép so sánh. Thực nghiệm cho thấy số phép so sánh và thời gian chạy của thuật toán đề xuất ít hơn đáng kể so với các giá trị số này của cách tiếp cận dựa trên thuật toán sắp xếp vun đống truyền thống. Kết quả này có được là do thuật toán đề xuất đã được thiết kế với kỹ thuật hạn chế các cặp được xét trong mỗi lần chọn ra cặp (x, y) có tổng bé nhất trong các cặp còn lại để đưa vào tập X + Y. Cấu trúc đống được sử dụng để lưu trữ và cập nhật các cặp được xét để chọn ra cặp nhỏ nhất giúp các thao tác trong mỗi bước chọn chỉ cần chạy trong thời gian O(log(n)) và cả thuật toán chạy trong thời gian $O(n^2log(n))$.

TÀI LIỆU THAM KHẢO/ REFERENCES

[1] E. Demaine, J. Erickson, and J. O'Rourke, "Problem 41: Sorting X + Y (Pairwise Sums)," The Open Problems Project, 20 August 2006. [Online]. Available: https://topp.openproblem.net/p41. [Accessed May 01, 2024].

[2] S. Skiena, "4.4 War Story: Give me a Ticket on an Airplane," in *The Algorithm Design Manual*, 2nd ed., Springer, 2008, pp. 118-120.

[3] D. A. Klip, "New Algorithms for Polynomial Multiplication," *SIAM Journal on Computing*, vol. 8, no. 3, pp. 326-343, 1979.

[4] D. S. Johnson, A. S. LaPaugh, and R. Y. Pinter, "Minimizing channel density by lateral shifting of components," in *the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, Arlington, Virginia, USA, 1994, pp. 122-131.

[5] A. Grønlund and S. Pettie, "Threesomes, Degenerates, and Love Triangles," in *IEEE 55th Annual Symposium on Foundations of Computer Science*, Philadelphia, PA, USA, 2014, pp. 621-630.

[6] M. T. El-Melegy and A. T. Kamal, "Color Image Processing Using Reduced Biquaternions With Application to Face Recognition in a PCA Framework," in *the IEEE International Conference on Computer Vision*, 2017, pp. 3039-3046.

[7] M. T. El-Melegy, A. T. Kamal, K. F. Hussain, and H. M. El-Hawary, "Classification by Principal Component Regression in the Real and Hypercomplex Domains," *Arab J. Sci. Eng.*, vol. 48, pp. 10099-10108, 2023.

[8] P. Kreitzberg, K. Lucke, and O. Serang, "Selection on $X_1 + X_2 + ... + X_m$ with layer-ordered heaps," *arXiv preprint*, 2020, doi: 10.48550/arXiv.1910.11993.

[9] O. Serang, "Optimally selecting the top k values from X + Y with layer-ordered heaps," *PeerJ Computer Science*, vol. 7, 2021, Art. no. e501.

[10] H. Kaplan, L. Kozma, O. Zamir, and U. Zwick, "Selection from heaps, row-sorted matrices and X + Y using soft heaps," *arXiv preprint*, 2018, doi: 10.48550/arXiv.1802.07041.

[11] J.-L. Lambert, "Sorting the sums $(x_i + y_j)$ in $O(n^2)$ comparisons," *Theoretical Computer Science*, vol. 103, no. 1, pp. 137-141, 1992.

[12] D. M. Kane, S. Lovett, and S. Moran, "Near-optimal linear decision trees for k-SUM and related problems," *arXiv preprint*, 2017, doi: 10.48550/arXiv.1705.01720.

[13] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, "Section 6 Heap sort," in *Introduction to Algorithms*, 4th ed., The MIT Press, 2022, pp. 161-172.