Mathematical and Software Engineering, vol. 4, no. 2 (2018), 24-27.
Varɛpsilon Ltd, http://varepsilon.com

Application of sorting algorithms for convex hull determination

Mihaela Todorova[1], Stoyan Kapralov[2], and Valentina Dyankova[1]

[1]Konstantin Preslavsky University of Shumen, Shoumen, Bulgaria,
emails: mihaela.todorova@shu.bg, v.dyankova@shu.bg

[2]Technical University of Gabrovo, Gabrovo, Bulgaria,
email: s.kapralov@gmail.com

Abstract

The proposed research explores the possibilities of applying some base algorithms for sorting to the process of finding a convex in order to optimize the time indicators of this process. A comparative analysis of the time characteristics has been performed using different time approaches in Graham's algorithm. The empirical results obtained have been used as a basis for building a pattern model of the process of finding a convex hull. It performs point sorting by a given criterion and finds a convex hull on a two-dimensional set of points. For this model, a visualization module has been developed that can be used as a learning environment in the courses of computing and complexity of algorithms.

Keywords: algorithms for sorting; convex hull; Graham's algorithm

1 Introduction

In modern computer systems, a number of basic and improved algorithms are used to organize and process large amounts of data. The process of rearranging objects in a particular order is known as sorting. In this sense sorting is a universal, basic activity, with wide application in different algorithms. One of the untraditional applications of sorting is in computational-geometric algorithms to find a convex hull of a set of points. The basis of these algorithms is the processing of a large number of multipoint points by a given criterion.

The definition of the convex hull finds interesting practical applications such as finger pointing [4], wireless sensor node recovery [5], virtual body digitalization [6], modelling of human solutions in games [7], machine learning [8] etc. For this reason, sorting is essential in the algorithms for finding a convex envelope. It is possible to sort out a large number of elements to take significant computing resources, and in this sense the question arises as to how exactly the type of sorting affects the efficiency and performance of the algorithm when used and applied. In the context of the discussed issues, the objectives of the present study are:

1. Suggest different approaches for integrating sorting algorithms into algorithms to find a convex envelope.

2. Perform studies of the influence of the sorting algorithm type in the process of finding a convex envelope and analyze and summarize the resulting experimental time results.

3. Using the optimal of the approaches studied, suggest an example algorithm on the basis of which a class for finding a convex envelope can be created.

2 Exposition

The essence of sorting as a process of rearranging a given set of objects in a particular order [3] determines the two underlying operations on which the complexity of each sorting algorithm depends. These are the operations of comparison and exchange of values. Since interest in the study is a sorting application in Graham's algorithm to find a convex shell, three of the main sorts will be examined. These are pyramidal sorting, bubble sorting and quick sorting.

The quick sort algorithm requires a time proportionally averaged over NlogN to sort the N element. The main drawback of the algorithm is that it is not stable and in worst-case uses N2 operations.

Bubble method is one of the most popular sorting algorithms. The average and worst case of this algorithm is O (n2) and is not used to sort large unordered datasets [1].

Heapsort is a kind of sorting algorithm by direct selection. With the complexity of $O(n \log n)$ it has the advantage of a more favorable worst case because he does not use many arrays or recursion.

Graham's algorithm for finding a convex envelope on multiple points consists of three basic steps. In its first step, using a sorting, there is a base point of the hull with minimum x and y coordinates. When searching for this point, all points in the set are sorted by the *y* coordinate, and if there are several points with a minimum y coordinate, they are sorted by the *x* coordinate. In the second step of the algorithm follows an angular range of points [2]. All points are sorted by the angle that forms the straight line passing through the base point and the point with the abscissa axis. This angular sort is replaced by checking the coordinates of the points. In its third step, the algorithm finds among the sorted points the ones that are from the shell and constructs it.

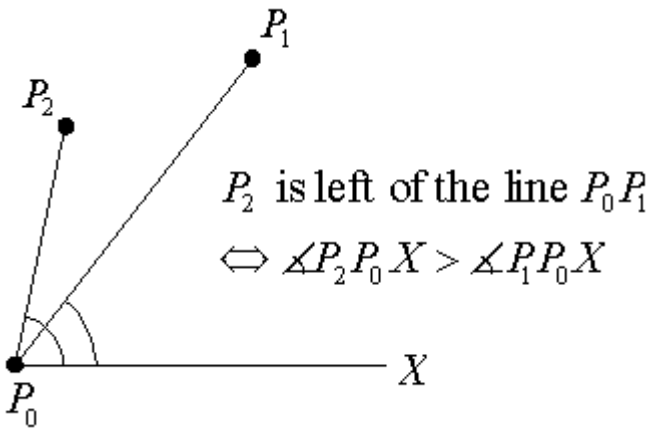

Figure 1: Angular sorting

For an empirical comparison of the timing characteristics when applying sorting in the Graham algorithm, the following approaches were considered:

1. Apply bubble method for the angular sorting.

When applying this algorithm, the base point of the set is detected and angularly sorted by the bubble method. The main modification in the algorithm is to compare the points of their location to their baseline and previous point. The array of points crawls in sequence and at each step compares the point a [*i*] and a [*i* +1]. If the point a [*i* +1] is located in the left half of the starting beam point and the point a [*i*], the algorithm continues with the check for points a [*i* +1] and a [*i* +2]. If the point a [*i* +1] is in the right half equal to this beam or lies on it, then the elements a [*i*] and a [*i*+1] change their positions.

2. Apply a quick sort order for angular sorting.

In the quick sort order to sort the stack of points first, a point *x* is selected as the last point in the array with which the other points are compared. After comparing with an auxiliary function, the goal is to the left of the x point in the array to be points which, compared to the base, make smaller angles with the abscissa and to the right of x points that make larger angles. Then, by recurs, the left and right parts of the array are sorted.

3. Apply a heapsort to find a base point.

When applying this sorting, there is a quick element with a minimum coordinate that stands at the top of the pyramid.

Table 1: Time performance

Input data	Bubble sort (seconds)	Quick sort (seconds)	Bubble sort and Heap sort (seconds)	Quick sort and Heap sort (seconds)
500	0.0146	0.0013	0.0123	0.0013
6000	1.686	0.035	1.676	0.0136
45000	96.514	0.5246	94.528	0.118
250000	/	4.539	/	0.782

Using a pyramid sorting in Graham's algorithm to find a base point, it is matched with a quick sort or a sorting method by arranging the points of the set relative to the angle.

In the Table 1 are given time results from the application of the three approaches for sorting the same data from the set of points. It can be seen that Graham's algorithm's time efficiency using a pyramidal point-to-point pairing in combination with a quick sorting of the remaining points gives the best time results from the approaches considered.

In the context of these results, Graham's sample algorithm, using a pyramidal and quick sort, and the basis on which to create a convex hull class can be proposed. The algorithm has the following form:

```
int grahamscan(Point A[], int n){
	sort(A,A+n,compy);
	int k=1;
	while(k<n && A[k-1].y==A[k].y) k++;
	if(k>1){
		sort(A,A+k,compx);
		swap(A[1],A[k-1]);
		if(k==n){
			if(A[0].x==A[n-1].x) return 1;
			return 2;
		}
		sort(A+k,A+n,compt);

	}
	else{
		sort(A+1,A+n,compt);
		k=2;
		while(k<n && dir(A[0],A[k-1],A[k])==0) k++;
		sort(A+1,A+k,compy);
		swap(A[1],A[k-1]);
	}
	int m=1;
	for(int i=k; i<n; i++){
		while (dir(A[m-1],A[m],A[i]) <= 0) m--;
		m++;
		swap(A[m],A[i]);
	}
	return m+1;
}
```

3 Conclusions

Applying a sort of algorithm to find a convex envelope can be explored in two main directions:

1. Approaches for adapting sorting algorithms when searching for convex envelope.
2. Ability to improve the efficiency and speed of algorithms for finding a convex hull.

The proposed algorithm successfully adapts the Gram's algorithm to look for a convex envelope, improving its performance and efficiency. For the application of sorting in other algorithms to find a convex envelope and their improvement, this is a good guideline for developing an optimal class that can be used to construct a convex envelope at a set of points

References

[1] Nakov, P., Dobrikov, P. (2018) *Programming = ++ Algorithms*, www.programirane.org.

[2] Cormen, T.H., Leiserson, Ch.E., Rivest, R.L., Stein, C. (2009) *Introduction to Algorithms*, Third Edition, MIT Press.

[3] Wirth, N., (1980) *Algorithms + Data Structures = Programs*, Prentice-Hall.

[4] Nagai, K., Sakabe, H., Ohka, M. (2017) Finger direction recognition toward human-and-robot cooperative tasks, International Symposium on Micro-NanoMechatronics and Human Science (MHS), 3-6 Dec. 2017, Nagoya, Japan, IEEE.

[5] Lakshmi, M., R. (2017) Modified Convex Hull Algorithm for Recovering Smashed Wireless Sensor Networks, International Journal of Scientific Research in Computer Science, Engineering and Information Technology, Volume 2, Issue 4, 495-502.

[6] Xu, Y., Hou, W. (2017) Calculation of operational domain of virtual maintenance based on convex hull algorithm, Second International Conference on Reliability Systems Engineering (ICRSE), 10-12 July 2017, Beijing, China.

[7] Tim Rach, Alexandra Kirsch. (2016) Modelling human problem solving with data from an online game. Cognitive Processing, Springer Verlag, 2016, 17 (4), pp.415-428.

[8] Pihera, J., Musliu, N. (2014) Application of Machine Learning to Algorithm Selection for TSP, 2014 IEEE 26th International Conference on Tools with Artificial Intelligence, 15 December 2014, Limassol, Cyprus.