Olympiads in Informatics, 2023, Vol. 17, 53–64

DOI: 10.15388/ioi.2023.05

Elementary Algorithms – Prefix Sum

László Gábor MENYHÁRT, László ZSAKÓ
Eötvös Loránd University, Budapest
e-mail: menyhart@inf.elte.hu, zsako@caesar.elte.hu

Abstract. In this paper we present a special problem, named as prefix sum, and its variations. We analyse the base problem more detailed and compare the possible solutions. Our students solved simi-lar problems in competitions, so we have statistics about their solutions. We offer this article for those readers who are interested in programming methodology, or who are teachers, and their students will be participated in competitions.

Keywords: systematic programming, teaching, measurement, prefix sum, cumulative sum.

1. Introduction

During teaching of programming, after the introduction of algorithmic structures (sequence, condition, loop), basic algorithms (programming patterns, type algorithms), such as e.g. summary, search, maxi-mum selection, sort, etc. are being dealt.

Problem-solving strategies appear in programming competitions (and in talent management) (e.g. greedy strategy, dynamic programming). However, the gap between the two is rarely filled. This was handled by J. Bentley's classic series of articles and book (Programming Pearls (Bentley, 1984)) from 1984, and more recently, for example, websites teaching algorithmization (Sannemo, 2018; ACM, 2023). Its first appearance was in 1954, in a magazine not directly on IT (Page, 1954).

In this article, we take a look at what the participants of Hungarian programming competitions do with the prefix sum (cumulative sum) task type. Interestingly, we are examining two very different age groups, in first students aged 13–16 can participate – Nemes Tihamér Online Programming Competition (Nemes, n.d.) (see later: Online), and in the other, university students who are studying computer science and just getting to know programming – Talent Search University Programming Competition (ELTE IK, n.d.) (see later: TUPC).

In this article, we do not intend to deal with advanced applications of the algorithm (balanced binary trees, Fenwick trees, ...).

2. Basic Task of Prefix Sum

Task: There is a series of numbers with `N` element in `X` and it contains both positive and negative numbers. Give the interval `[a, b]` of series where the sum of the elements is maximal! We must define such an a and b indexes for which the following condition is satisfied:

$$1 \leq a \leq b \leq N$$

and

$$\forall p, q\ (1 \leq p \leq q \leq N)\colon \sum_{i=a}^{b} X_i \geq \sum_{i=p}^{q} X_i$$

Solution: The basic solution calculates the sum of the numbers for each interval, and then selects the maximum for them:

```
sum(X,i,j):
  S:=0
  For k=i to j
    S:=S+X[k]
  End for
  sum:=S
End function.

BasicSolution(N,X,a,b,max):
  max:=-∞
  For i=1 to N
    For j=i to N
      s:=sum(X,i,j)
      If s>max then max:=s; a:=i; b:=j
    End for
  End for
End function.
```

Due to the three loops, its running time is proportional to N^3. A frequently used idea is to first calculate not what the task asks for, but something simpler, from which it is easy to get the solution to the task.

Let's calculate the sum of the first `i` element of the series in the vector `s`.

$$\forall i\ (1 \leq i \leq N)\colon s(i) = \sum_{j=1}^{i} X_j$$

which, using a recursive relation, is simpler:

$$\forall i\ (1 \leq i \leq N)\colon s(i) = s(i-1) + X(i), \qquad s(0) = 0$$

Then, the sum of the elements of an interval can be expressed as follows (Fig. 1):

$$\sum_{i=a}^{b} X_i = s(b) - s(a-1)$$

Thus, the following algorithm – the method of cumulative summation – takes a step proportional to N^2.

```
Cumulative sum(N,X,a,b,max):
  s[0]:=0
  For i=1 to N
    s[i]:=s[i-1]+X[i]
  End for
  max:=-∞
  For i=1 to N
    For j=i to N
      If s[j]-s[i-1]>max then max:=s[j]-s[i-1]; a:=i; b:=j
    End for
  End for
End function.
```

3. Variations of Prefix Sum

This strategy can be used not only for summation (serial calculation), but also for other types of tasks. Basically, in cases where operations are associative (Blelloch, 1990), there is an "inverse" (Sannemo, 2018), so

$$f(a,b) = f^{-1}(f(1,b), f(1,a-1)).$$

Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	
Series		3	5	1	8	2	4	
prefix sum	0	3	8	9	17	19	23	
sum(1,6)	0	3	8	9	17	19	23	
sum(1,2)	0	3	8	9	17	19	23	
sum(3,5)	0	3	8	9	17	19	23	19-8=11

Fig. 1. prefix sum.

For example

$$\sum_{i=a}^{b} X_i = \sum_{i=1}^{b} X_i - \sum_{i=1}^{a-1} X_i$$

or

$$X_a * X_{a+1} * \ldots * X_b = \frac{X_1 * X_2 * \ldots * X_b}{X_1 * X_2 * \ldots * X_{a-1}}$$

assuming that none of the values is `0`.

However, the maximum selection task for the interval `[a, b]` is not always interesting for the above, for example, for the sum it is uninteresting if all numbers are positive, and for the product, if all numbers are greater than `1`.

Often the task is not maximum selection, but answering many questions for different intervals (Yao and Miaoh, 1990; USACO, 2015). In another type, we are looking for a certain type of sum, in the competition problem (USACO, 2016), for example, the longest series, where the sum of the numbers is divisible by `7`.

Counting can also make sense for the item, but then either the length of the interval or some other property must be modified.

They can be:

- Let's give a sequence of at most K length, in which the number of elements with the given property is maximal!
- Let's give at most K long part of a series, in which the number of elements with the given property is maximal and the two extreme elements also have the given property!
- Let's give the longest part of a series in which at least half of the elements have a given property and the two extreme elements!

In many tasks, the method is included as a supplement, moreover, only if certain prerequisites are met.

Task: A random number generator generates numbers between `0` and `M-1`. We received the first `N` random numbers produced by the generator. Write a program to check the generator, it calculates for `K` pieces of interval `[A,B]`, how many numbers between `0` and `M-1` do not occur in the given interval! The value of `K` is much greater than the value of `M`!

Solution: We can write a not so naive solution for it (it solves the counting by indexing):

```
None(N,X,K):
  For i=1 to K
    In: A,B
    number[]:=(0,...,0)
    For j=A to B
      number[X[j]]:=number[X[j]]+1
    End for
```

```
    C:=0
    For j=0 to M-1
      If number[j]=0 then C:=C+1
    End for
    Out: C
  End for
End function.
```

According to this, its running time is proportional to `K*(N+M)`.

Cumulative summation first calculates in a matrix how many `j` values are in the first random number `i`, and then calculates the result from this:

```
None(N,X,K):
  number[0]:=(0,...,0)
  For i=1 to N
    number[i]:=number[i-1]; number[i, X[i]]:=number[i, X[i]]+1
  End for
  For i=1 to K
    In: A, B
    C:=0
    For j=0 to M-1
      If number[b, j]-number[a-1, j]=0 then C:=C+1
    End for
    Out: C
  End for
End function.
```

The first loop is N*M, and the second is K*M, even in the case of slow implementation of array evaluation. According to the task description, K is much larger than M, so K*N+K*M is greater than N*M+K*M, and this is true if K>M, as stated in the task description.

4. Add an Extra Step

We can modify the task with extra steps. For instance, the solution of next task is more efficient with order.

Task: At an event, the number of male and female visitors is recorded for each day. Create a program that gives the period during which the total number of men and women was the closest to each other!

Solution: We can create the naïve solution when all time periods are calculated. Let `x[i]` be `i.` the difference in the number of men and women per day! Running time $O(N^3)$.

```
  min:=+∞
  For i=1 to N
    For j=i to N
      S:=0
      For k=i to j
```

```
      S:=S+X[k]
    End For
    If |S|<min then mink:=i; minv:=j; min:=|S|
  End For
End For
```

In the second version we calculate prefix amounts, and then take the smallest one from their difference. Running time $O(N^2)$.

```
s[0]:=0
For i=1 to N
  s[i]:=s[i-1]+X[i]
End For
min:= +∞
For i=1 to N
  For j=i to N
    If |s[j]-s[i-1]|<min
      then mink:=i; minv:=j; min:=|s[j]-s[i-1]|
  End For
End For
```

The optimal solution uses the difference between arranged and adjacent ones. The difference between two values in s is the smallest if they are closest to each other in value. Let's arrange them in a row, then take the one with the smallest difference from the adjacent ones! Running time $O(N*\log_2 N)$.

```
s[0].db:=0; s[0].ind:=0
For i=1 to N
  s[i].db:=s[i-1].db+X[i]; s[i].ind:=i
End For
Order(s,0..N)
mini:=1
For i=1 to N-1
  If s[i+1].db-s[i].db<s[mini+1].db-s[mini].db then mini:=i
End For
min:=s[mini+1,1]-s[mini,1]
mink:=less(s[mini].ind,s[mini+1].ind)+1
minv:=greater(s[mini].ind,s[mini+1].ind)
```

Based on the test cases, we managed to achieve different scores for the different solution methods, so we can determine the distribution of solution types based on the scores achieved by the competitors (Fig. 2).

optimal	71–100
prefix sum	31–70
naïve	0–30

Fig. 2. Scoring of different solutions.

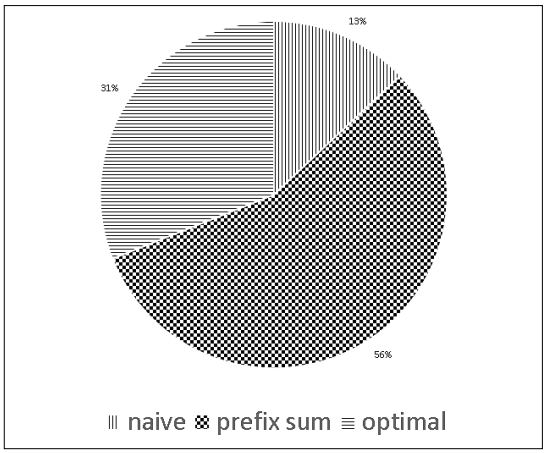

Fig. 3. Distribution of solution types.

Different scores within each group may be due to good or bad handling of special cases.

Fig. 3 shows the distribution of scores between each type of solution.

5. Mathematics and Matrices with Prefix Sum

A prefix sum problem can be generalized not only to vectors, but also to other structures, for example to a matrix as you can see in the following:

Task: A farmer wants to buy a rectangular plot of land in an `N×M` rectangle. He knows about each piece of land that can be bought, how much he would profit or lose if he cultivated it. Enter the rectangle on which the greatest profit can be achieved!

Solution: In the elementary solution, the sum of the elements of each `(P,Q)` upper-left and `(R,S)` lower-right sub-rectangle would have to be calculated, which would result in `6` loops, and then we could take their maximum (Fig. 4).

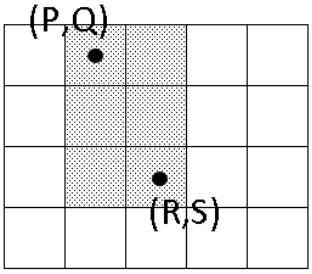

Fig. 4. The task.

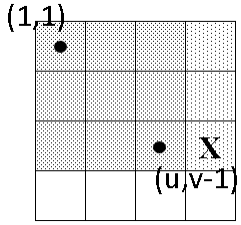

Fig. 5. Calculation of E.

Let's try to set a partial goal: calculate the value of the `(1,1)` upper left and `(u, v)` lower right corner tiles and store it in `E(u, v)`!

`X = value of densely dotted rectangle + value of rarely dotted rectangle` (Fig. 5)

```
E[1..N,0]:=(0,...0)
For v=1 to M
  x:=0
  For u=1 to N
    x:=x+T[u,v]
  End for
  E[u,v]:=E[u,v-1]+x
End for
```

Based on these, the sum of the values of the `(i, j)` upper left and `(u, v)` lower right rectangles is: `value(E,i,j,u,v)=E[u,v]-E[u,j-1]-E[i-1,v]+E[i-1-,j-1]`. See the same on Fig. 6!

	…	**j-1**	**j**	…	**v**	…
…						
i-1	…	$+E_{i-1,j-1}$	$E_{i-1,j}$	…	$-E_{i-1,v}$	…
i	…	$E_{i,j-1}$	$E_{i,j}$	…	$E_{i,v}$	…
…						
u	…	$-E_{u,j-1}$	$E_{u,j}$	…	$E_{u,v}$	…
…						

Fig. 6. Calculation of result.

We write here the algorithm only so that we can modify it in the next step:

```
max:=-∞
For i=1 to N
  For j=1 to M
    For k=i to N
      For l=j to M
        o:=value(E,i,j,k,l)
        If o>max then P,Q,R,S:=i,j,k,l; max:=o
      End for
    End for
  End for
End for
```

The `4` loops are clearly visible, i.e. if `N` and `M` are of the same order of magnitude, then the running time is proportional to $O(N^4)$, to the fourth power of `N`.

In the competitions, we modified the task in the same way for both age groups, we fixed the area of the rectangle to be selected, i.e. we searched for four values for which there was a new condition that `(R-P+1)*(S-Q+1)=A`, for a given `A` constant, and the number of elements of some type had to be specified instead of the sum.

One of the tasks of the 3rd round of the competition held in 2017/18 was as follows: "In a rectangular area, we know the altitude above sea level of `N*M` points. Points that are larger than their four neighbours are called vertices. There are certainly no peaks at the edges of the area. Write a program that gives you a rectangular area containing exactly `A` points, with as many vertices as possible!"

A significant part of the competitors, as expected based on the previous ones, gave the naive solution, with `6` loops. A significant part of those competitors who knew the method examined the size of the area within the above `4` loops:

```
If (k-i+1)*(l-j+1)=A then
    o:=value(E,i,j,k,l)
    If o>max than P,Q,R,S:=i,j,k,l; max:=o
```

Those who were a little more skilled could have saved one more loop and calculated the value of the variable l only for k values where A divided by `(k-i+1)`, but we hardly found such a solution.

```
max:=-∞
For i=1 to N
  For j=1 to M
    For k=i to N
      If A mod (k-i+1)=0 then
        o:=value(E,i,j,k,j+A div k-1)
        If o>max then P,Q,R,S:=i,j,k, j+A div k-1; max:=o
      End if
    End for
  End for
End for
```

In the most effective solution, the divisors of A can be calculated in square_root(A)[1] steps into a vector D with C elements, since the area can be calculated as the product of two sides.

Thus, the expected solution is $O(N^2C)$, taking advantage of the fact that D[k]*D[C-k+1]=A:

```
max:=-∞
For i=1 to N
  For j=1 to M
    For k=1 to C
      o:=value(E,i,j,i+D[k]-1,j+D[C-k+1]-1)
      If o>max then P,Q,R,S:=i,j,i+D[k]-1,j+D[C-k+1]-1; max:=o
    End for
  End for
End for
```

Based on the test cases, we managed to achieve different scores for the different solution methods, so we can determine the distribution of solution types based on the scores achieved by the competitors (Fig. 7).

Different scores within each group may be due to good or bad handling of special cases.

The university students use their more mathematical knowledge to score more points, and that they are moving from the naive and cumulative solution to a more efficient solution. They didn't even have a naive and simple cumulative solution. On the other hand, these together hardly reached the 50% rate in the 13–16 age group. However, the best competitors also found the optimal solution (Fig. 8).

6. Conclusion

The basic idea of the cumulative summation is therefore: Instead of a value calculated on an arbitrary subseries, we should only calculate a value for that subseries whose begin-

optimal, only with divider k, calculation of l from k	92–100
only with divider k, only with divider l	71–91
calculation of k from l	55–70
simple cumulative	41–54
naive	16–40
good with luck	0–15

Fig. 7. scoring of different solutions.

[1] It's enough to determine all divider till square root of A, because if x is divider of A, A/x is also divider.

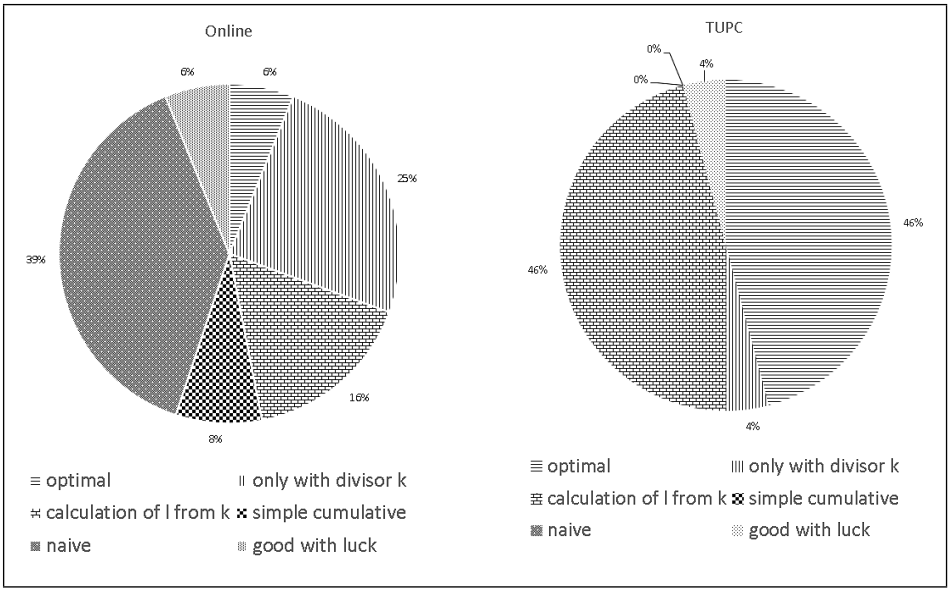

Fig. 8. Distribution of solution types.

ning is the beginning of the whole series, and then express the value calculated on the subseries from these!

The same principle can also be applied to calculation tasks instead of summation, i.e. the operation can be any associative operation whose inverse can be calculated.

Based on our experience in the competition, students learn the basic method easily, but they do not figure it out by themselves – typically, the younger we look at, the more the proportion of naive, slow solution makers increases. Only some of them notice the small modifications, e.g. when such sums need to be calculated both from the front and from the back, or another operation must be used instead of summation.

It is important to note that naive solutions include summation and maximum selection for most tasks, but these elementary algorithms (programming theorems) are also needed to write effective, cumulative summation. In other words, we are not talking about new algorithms, but we must think differently when applying the usual algorithms!

Mathematical considerations, as we saw in the last task, cause problems for students who are not strong in mathematics. And the math was just that if an integer is the product of two integers, then both numbers must divide the product – so it is necessary to produce the divisors. It can also be seen from the results that a significantly higher proportion of first semester university students solved the task with the maximum score.

Because of the above, we believe that in teaching programming, we should deal with the above basic method (avoiding all more complicated data structures). In this, teachers currently receive little help (Szlávi and Zsakó, n.d.), because the vast majority of scientific articles are about advanced solutions using different trees.

References

ACM (2023). Category: Teaching kids programming. *Algorithms, Blockchain and Cloud.* (Accessed on: 2023.05.26.) https://helloacm.com/category/teaching-kids-programming/

Bentley, J. (1984). Programming Pearls – Algorithm Design Techniques. *Comm. ACM*, 27(9), 865–871.

Blelloch, G.E. (1990). *Prefix Sums and Their Applications*. Tech. Rep. CMU-CS-90-190. School of Computer Science, Carnegie Mellon University, Pittsburgh. (Accessed on: 2023.05.26.) https://www.cs.cmu.edu/~guyb/papers/Ble93.pdf

ELTE IK (n.d.).Talent Search University Programming Competition. *Informatikai versenyek.* (Accessed on: 2023.05.26.) http://versenyek.inf.elte.hu/versenyek/tehetsegkutato-egyetemi-programozasi-verseny

Nemes, T. (n.d.). *Online Programming Competition*. (Accessed on: 2023.05.26.) http://tehetseg.inf.elte.hu/nemes-online/index.html

Page, E.S. (1954). Continuous Inspection Schemes. *Biometrika*, 41(1/2), 100–115. (Accessed on: 2023.05.26.) https://www.jstor.org/stable/2333009?origin=JSTOR-pdf&seq=1#metadata_info_tab_contents

Sannemo, J. (2018). Principles of Algorithmic Problem Solving. (Accessed on: 2023.05.26.) https://usaco.guide/PAPS.pdf#page=207

Szlávi, P., Zsakó, L. (n.d.) Elemi algoritmusok (a programozási tételek után). ELTE Informatikai Kar, Budapest. (Accessed on: 2023.05.26.) http://tehetseg.inf.elte.hu/szakkorefop2017/pdf/elteikszakkor_elemi_algoritmusok.pdf

USACO (2015). *USACO 2015 December Contest, Silver, Problem 3. Breed Counting*. USA Computing Olympiad. (Accessed on: 2023.05.26.) http://www.usaco.org/index.php?page=viewproblem2&cpid=572

USACO (2016). *USACO 2016 January Contest, Silver Problem 2. Subsequences Summing to Sevens*. USA Computing Olympiad. (Accessed on: 2023.05.26.) http://www.usaco.org/index.php?page=viewproblem2&cpid=595

Yao, D., Miaoh, D. (n.d.). Introduction to Prefix Sums. (Accessed on: 2023.05.26.) https://usaco.guide/silver/prefix-sums?lang=cpp

L.G. Menyhárt is assistant professor at Department of Media & Educational Informatics, Faculty of Informatics, Eötvös Loránd University in Hungary. Since 2003 he has been involved in task creation of programming competitions in Hungary. His research interest includes teaching algorithms and data structures; didactics of informatics; methodology of programming in education; teaching programming languages; talent management.

L. Zsakó is a professor at Department of Media & Educational Informatics, Faculty of Informatics, Eötvös Loránd University in Hungary. Since 1990 he has been involved in organizing of programming competitions in Hungary, including the CEOI. He has been a deputy leader for the Hungarian team at IOI since 1989. His research interest includes teaching algorithms and data structures; didactics of informatics; methodology of programming in education; teaching programming languages; talent management. He has authored more than 68 vocational and textbooks, some 200 technical papers and conference presentations.