Začeto dne	torek, 2. april 2019, 09:23
Stanje	Zaključeno
Dokončano dne	nedelja, 7. april 2019, 22:51
Porabljeni čas	5 dni 13 ure
Točke	1,00/1,00
Ocena	**10,00** od možne ocene 10,00 (**100**%)

Vprašanje **1**

Pravilno

Ocena 1,00 od 1,00

Napiši program, ki zna iz standardnega vhoda prebrati zaporedje števil in poiskati strnjeno podzaporedje z največjo vsoto elementov. Pri reševanju naloge uporabi deli in vladaj algoritem, ki za delitev problema izrablja dejstvo, da se strnjeno zaporedje z največjo vsoto nahaja bodisi v levi polovici zaporedja bodisi v desni polovici zaporedja bodisi vsebuje vsaj en element iz leve in vsaj en element iz desne polovice zaporedja. Koraki, ki jih algoritem izvaja, so naslednji:

1. vhodno tabelo razdeli na levo in desno polovico,
2. v obeh delih tabele poišči podzaporedje z največjo vsoto,
3. poišči največjo vsoto zaporedja, ki vsebuje vsaj en element iz leve in in vsaj en element iz desne polovice tabele,
4. vrni največjo vsoto izmed treh izračunanih.

Na zaporedju [2, 3, 1, -1, 4, -5, 3] bi algoritem izvedel naslednje korake:

1. zaporedje bi razdelil na [2, 3, 1, -1] in [4, -5, 3],
2. v levi polovici bi našel vsoto vsoto 6 (podzaporedje [2, 3, 1]) in na desni 4 (podzaporedje [4]),
3. vsota podzaporedja, ki se razteza čez obe polovici, bi bila v tem primeru 9 (podzaporedje [2, 3, 1, -1, 4]),
4. kot rezultat vrne max(6, 4, 9) = 9.

Iskanje vsote strnjenega podzaporedja, ki vsebuje elemente iz leve in desne polovice zaporedja, izvedemo tako, da v kandidatno podzaporedje dodamo zadnji element leve polovice in prvi element desne polovice zaporedja, ki ga nato širimo levo in desno, med širjenjem pa si zapomnimo največjo doseženo vrednost vsote. Za zgornji primer bi algoritem tako izvedel naslednje korake:

1. začetno vsoto podzaporedja nastavimo na 3 (vsota elementov -1 iz leve polovice in 4 iz desne polovice) in jo obenem uporabimo kot trenutno največjo vsoto,
2. ko dodamo 1, vsota kandidatnega zaporedja naraste na 4, zato popravimo tudi največjo vsoto na 4,
3. ko dodamo 3, vsota kandidatnega zaporedja naraste na 7, zato popravimo tudi največjo vsoto na 7,
4. ko dodamo 2, vsota kandidatnega zaporedja naraste na 9, zato popravimo tudi največjo vsoto na 9,
5. ker smo prišli do levega roba, vsoto kandidatnega zaporedja nastavimo na trenutno največjo vsoto,
6. ko dodamo -5, vsota kandidatnega zaporedja pade na 4, zato največje vsote ne popravljamo,
7. ko dodamo 3, vsota naraste na 7, a ker je še vedno manjša od trenutno največje vsote, le-te ne popravimo.
8. končni rezultat je torej 9.

Program naj na standardni izhod ob vsaki določitvi največje vsote izpiše zaporedje ter izračunano največjo vsoto podzaporedja.

For example:

Input	Result

`2 3 1 -1 4 -5 3`	`[2]: 2`
	`[3]: 3`
	`[2, 3]: 5`
	`[1]: 1`
	`[-1]: -1`
	`[1, -1]: 1`
	`[2, 3, 1, -1]: 6`
	`[4]: 4`
	`[-5]: -5`
	`[4, -5]: 4`
	`[3]: 3`
	`[4, -5, 3]: 4`
	`[2, 3, 1, -1, 4, -5, 3]: 9`

`-9 4 3 -5`	`[-9]: -9`
	`[4]: 4`
	`[-9, 4]: 4`
	`[3]: 3`
	`[-5]: -5`
	`[3, -5]: 3`
	`[-9, 4, 3, -5]: 7`

Answer: (penalty regime: 0 %)

```
import java.util.*;

public class Izziv5 {

    private static Scanner sc = new Scanner(System.in);
    private static int velikostSeznam;

    public static void main(String[] args) {
        List<Integer> seznam = new ArrayList<>(0);
        while(sc.hasNext())
            seznam.add (sc.nextInt());
        najdi(seznam);

    }
```

```
15
16      private static int najdi(List<Integer> seznam) {
17          velikostSeznam = seznam.size();
18          if(velikostSeznam == 1)
19              return izpis0(seznam);
20
21              List<Integer> l = new ArrayList<>(0);
```

	Input	Expected	Got	
✔	2 3 1 -1 4 -5 3	[2]: 2 [3]: 3 [2, 3]: 5 [1]: 1 [-1]: -1 [1, -1]: 1 [2, 3, 1, -1]: 6 [4]: 4 [-5]: -5 [4, -5]: 4 [3]: 3 [4, -5, 3]: 4 [2, 3, 1, -1, 4, -5, 3]: 9	[2]: 2 [3]: 3 [2, 3]: 5 [1]: 1 [-1]: -1 [1, -1]: 1 [2, 3, 1, -1]: 6 [4]: 4 [-5]: -5 [4, -5]: 4 [3]: 3 [4, -5, 3]: 4 [2, 3, 1, -1, 4, -5, 3]: 9	✔
✔	-9 4 3 -5	[-9]: -9 [4]: 4 [-9, 4]: 4 [3]: 3 [-5]: -5 [3, -5]: 3 [-9, 4, 3, -5]: 7	[-9]: -9 [4]: 4 [-9, 4]: 4 [3]: 3 [-5]: -5 [3, -5]: 3 [-9, 4, 3, -5]: 7	✔
✔	-6 -6 -7 -1 0	[-6]: -6 [-6]: -6 [-6, -6]: -6 [-7]: -7 [-6, -6, -7]: -6 [-1]: -1 [0]: 0 [-1, 0]: 0 [-6, -6, -7, -1, 0]: 0	[-6]: -6 [-6]: -6 [-6, -6]: -6 [-7]: -7 [-6, -6, -7]: -6 [-1]: -1 [0]: 0 [-1, 0]: 0 [-6, -6, -7, -1, 0]: 0	✔
✔	4 0 -6 9 3 -2	[4]: 4 [0]: 0 [4, 0]: 4 [-6]: -6 [4, 0, -6]: 4 [9]: 9 [3]: 3 [9, 3]: 12 [-2]: -2 [9, 3, -2]: 12 [4, 0, -6, 9, 3, -2]: 12	[4]: 4 [0]: 0 [4, 0]: 4 [-6]: -6 [4, 0, -6]: 4 [9]: 9 [3]: 3 [9, 3]: 12 [-2]: -2 [9, 3, -2]: 12 [4, 0, -6, 9, 3, -2]: 12	✔
✔	-6 -6 3 4 -6 7 -9 -3 -6 -8	[-6]: -6 [-6]: -6 [-6, -6]: -6 [3]: 3 [-6, -6, 3]: 3 [4]: 4 [-6]: -6 [4, -6]: 4 [-6, -6, 3, 4, -6]: 7 [7]: 7 [-9]: -9 [7, -9]: 7 [-3]: -3 [7, -9, -3]: 7 [-6]: -6 [-8]: -8 [-6, -8]: -6 [7, -9, -3, -6, -8]: 7 [-6, -6, 3, 4, -6, 7, -9, -3, -6, -8]: 8	[-6]: -6 [-6]: -6 [-6, -6]: -6 [3]: 3 [-6, -6, 3]: 3 [4]: 4 [-6]: -6 [4, -6]: 4 [-6, -6, 3, 4, -6]: 7 [7]: 7 [-9]: -9 [7, -9]: 7 [-3]: -3 [7, -9, -3]: 7 [-6]: -6 [-8]: -8 [-6, -8]: -6 [7, -9, -3, -6, -8]: 7 [-6, -6, 3, 4, -6, 7, -9, -3, -6, -8]: 8	✔

	Input	Expected	Got	
✔	-3 7 1 -1 8 -5 -6 2 -2 -2 -7 -2 -2 7 8 0 2 -8 -4 -8	[-3]: -3 [7]: 7 [-3, 7]: 7 [1]: 1 [-3, 7, 1]: 8 [-1]: -1 [8]: 8 [-1, 8]: 8 [-3, 7, 1, -1, 8]: 15 [-5]: -5 [-6]: -6 [-5, -6]: -5 [2]: 2 [-5, -6, 2]: 2 [-2]: -2 [-2]: -2 [-2, -2]: -2 [-5, -6, 2, -2, -2]: 2 [-3, 7, 1, -1, 8, -5, -6, 2, -2, -2]: 15 [-7]: -7 [-2]: -2 [-7, -2]: -2 [-2]: -2 [-7, -2, -2]: -2 [7]: 7 [8]: 8 [7, 8]: 15 [-7, -2, -2, 7, 8]: 15 [0]: 0 [2]: 2 [0, 2]: 2 [-8]: -8 [0, 2, -8]: 2 [-4]: -4 [-8]: -8 [-4, -8]: -4 [0, 2, -8, -4, -8]: 2 [-7, -2, -2, 7, 8, 0, 2, -8, -4, -8]: 17 [-3, 7, 1, -1, 8, -5, -6, 2, -2, -2, -7, -2, -2, 7, 8, 0, 2, -8, -4, -8]: 17	[-3]: -3 [7]: 7 [-3, 7]: 7 [1]: 1 [-3, 7, 1]: 8 [-1]: -1 [8]: 8 [-1, 8]: 8 [-3, 7, 1, -1, 8]: 15 [-5]: -5 [-6]: -6 [-5, -6]: -5 [2]: 2 [-5, -6, 2]: 2 [-2]: -2 [-2]: -2 [-2, -2]: -2 [-5, -6, 2, -2, -2]: 2 [-3, 7, 1, -1, 8, -5, -6, 2, -2, -2]: 15 [-7]: -7 [-2]: -2 [-7, -2]: -2 [-2]: -2 [-7, -2, -2]: -2 [7]: 7 [8]: 8 [7, 8]: 15 [-7, -2, -2, 7, 8]: 15 [0]: 0 [2]: 2 [0, 2]: 2 [-8]: -8 [0, 2, -8]: 2 [-4]: -4 [-8]: -8 [-4, -8]: -4 [0, 2, -8, -4, -8]: 2 [-7, -2, -2, 7, 8, 0, 2, -8, -4, -8]: 17 [-3, 7, 1, -1, 8, -5, -6, 2, -2, -2, -7, -2, -2, 7, 8, 0, 2, -8, -4, -8]: 17	✔

Passed all tests! ✔

Question author's solution:

```
import java.util.Scanner;
import java.util.ArrayList;

public class Izziv5 {
  private static void print(int[] input, int l, int r, int sum) {
    System.out.print("[" + input[l]);
    for (int i = l + 1; i <= r; i++)
      System.out.print(", " + input[i]);
    System.out.println("]: " + sum);
  }

  private static int maxSubSum(int[] input, int l, int r) {
    if (l == r) {
      print(input, l, r, input[l]);
      return input[l];
    }

    int m = (l + r) / 2;

    int tmpSum = input[m];
    int lSum = tmpSum;
    for (int i = m - 1; i >= l; i--) {
      tmpSum += input[i];
      if (tmpSum > lSum) {
        lSum = tmpSum;
      }
    }

    tmpSum = input[m + 1];
    int rSum = tmpSum;
    for (int i = m + 2; i <= r; i++) {
      tmpSum += input[i];
      if (tmpSum > rSum) {
        rSum = tmpSum;
      }
    }

    int result = Math.max(Math.max(maxSubSum(input, l, m),
                                   maxSubSum(input, m + 1, r)),
                          lSum + rSum);
    print(input, l, r, result);

    return result;
  }

  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    ArrayList<Integer> input = new ArrayList<>();

    while (sc.hasNextInt()) {
      input.add(sc.nextInt());
    }
    int[] table = input.stream().mapToInt(Integer::intValue).toArray();

    maxSubSum(table, 0, table.length - 1);
  }
}
```

Pravilno

Marks for this submission: 1,00/1,00.

◄ Kviz 5 Skok na... Izziv 6 (Strassen) ►