Slovní popis:

Jednotlivé stánky na akci můžeme reprezentovat pomocí pole, ve kterém jsou jednotlivé prvky reprezentovány cenami. Naším úkolem je najít nejdelší posloupnost prvků pole, jejichž součet je menší než číslo *k* reprezentující počet korun.

Úlohu je možné řešit online. Nemusíme tedy načítat celé pole a až poté spouštět program. Ačkoliv může tento způsob mírně zpomalit program kvůli opakovanému získávání čísel ze vstupu, je mnohem šetrnější k paměťové složitosti.

Na začátku deklarujeme proměnné:

- currentCount – Počet dortíků, které by mohl Radek sníst, kdyby si vybral aktuální posloupnost čísel
- lastMaxStartingPosition – Počáteční index posloupnosti s nejvyšším známým počtem dortíků
- currentSum – součet cen dortíků aktuální posloupnosti
- i – konečný index aktuální posloupnosti
- helpQueue – pomocná fronta obsahující všechna aktuální čísla posloupnosti.

Program iteruje přes všechny prvky na vstupu, kdy na počátku uloží aktuální cenu dortíku do pomocné fronty. Poté otestuje, zda by tato cena sečtená s aktuálním součtem byla nižší než počet Radkových peněz. Pokud ano, přičte k aktuálnímu součtu cenu aktuálního dortíku, zapíše index startovní pozice posloupnosti a následně zvětší proměnnou currentCount o 1. Pokud ne, je od aktuálního součtu odečtena hodnota začátku aktuální posloupnosti a je přičtena hodnota aktuálního dortu. Nakonec vrátíme currentCount a lastMaxStartingPosition.

Zdůvodnění správnosti:

Jakmile má alespoň *m* po sobě jdoucích prvků součet menší než *k*, poté je jasné, že číslo *d* musí být alespoň *m.* Není tedy potřeba proměnnou currentCount nikdy snižovat. Program funguje na principu, že je sledováno okno o délce alespoň *m* a pokud je součet prvků v tomto okně menší než *k,* vyzkouší se přidání dalšího prvku. V případě, že to možné je, zvýší se číslo *m* o 1 a každá další posloupnost splňující podmínky musí být delší, aby bylo třeba jí věnovat pozornost.

Zároveň platí, že pokud okno posouváme, stačí odečíst první prvek posloupnosti a přičíst nový, jelikož součet ostatních prvků je již obsažen v původním součtu.

Pseudokód:

```
int currentCount, lastMaxStartingPosition, currentSum, i = 1;
Queue helpQueue = new Queue();
for (; i <= n; i++) {
  int currentCakePrice = next();
  helpQueue.enqueue(currentCakePrice);
  if ((currentSum + currentCakePrice) <= k) {
    currentSum += currentCakePrice;
    lastMaxStartingPosition = i - currentCount;
    currentCount ++;
  } else {
    currentSum = currentSum – helpQueue.dequeue() + currentCakePrice;
  }
}
return currentCount, lastMaxStartingPosition;
```

*dequeue zároveň odstraní prvek z fronty. V případě offline řešení: cakePrices[i - currLength]

Asymptotická složitost:

Program je tvořen jedním cyklem procházejícím vstupní data, kdy se při každé iterace posuneme na další prvek. Asymptotická složitost je linerátní, tedy **O(n)**

Paměťová složitost:

Pokud je úloha řešená online, paměť je zabrána pouze pomocnými proměnnými a pomocnou frontou, která má velikost nejvýše délky čísla *d* (vrácený počet dortíků). Paměťová složitost je tedy O(1+d)=**O(d)**