# Vraag 1

 *(Aanbevolen tijd: 20 min.)*

Je hebt een programma die de prijstoename en -afname voorspelt van een aandeel in de loop van de tijd. Op basis van die gegevens wil je éénmalig dit aandeel kopen op een bepaalde dag en vervolgens later verkopen zodat je winst maximaal is.

Opgave: gegeven een array van integers, bepaal de maximale som van alle mogelijke deelsequenties (m.a.w. deelreeks van aansluitende waarden in de array).



Figuur 1: de maximale subarray is aangeduid in dit voorbeeld.

Implementeer de methode “**stocks**” in de meegegeven klasse **Stocks** in package“**vraag1**”. Geef een **int** terug die de grootste som teruggeeft. De oplossing wordt automatisch voor jou gecontroleerd als je op “run” klikt.

# Vraag 2

*(Aanbevolen tijd: 100 min.)*

In dit deel gaan we een simpele versie van het spel “Tron” implementeren in package “**vraag2**”. Dit spel lijkt op “Slither.io”: twee spelers beschrijven een pad van verschillende kleuren. Wanneer een van de spelers zijn eigen pad of het pad van de tegenstander raakt, is deze geëlimineerd.

### Deel 1

Maak eerst de klasse **Punt**, die bevat x en y coördinaten elk van het type **int.** Maak dan de klasse **Speler**. Een Speler heeft een positie gegeven door een **Punt**, en bevat een ook lijst van **Punt**en genaamd “**lijnreeks**” die het getraceerde pad van de Speler bijhoudt. Die lijst stelt **N** aan elkaar verbonden lijnstukken voor, gebruik makende van **N+1** **Punt**en (zie Figuur 2)

Maak en implementeer de volgende methodes in **Speler**:

1. **replacepoint(Punt p):** vervangt het laatste Punt in **lijnreeks** door het nieuwe meegegeven Punt.
2. De methodes **isHorizontaal(int idx)** en **isVerticaal(int idx)** moeten een **boolean** teruggeven**;** deze geven aan of het lijnstuk met index “idx” respectievelijk horizontaal of verticaal georiënteerd is.



Figuur 2: Lijst van N+1 punten die N lijnstukken voorstelt

Vervolgens willen we bepalen of een meegegeven Punt ligt op de lijn. Voor elk paar openvolgende punten en (overeenkomstig met elk lijnstuk) moeten we dus controleren of er voldaan wordt aan de volgende vergelijking:

 waar

Dit kunnen we achterhalen door te projecteren op het lijnstuk gevormd door en (resulterend in ), en vervolgens de afstand tussen en te meten. We kunnen te waarde van het geprojecteerde punt achterhalen gebruik makende van de volgende formule:

Invullen in de vorige vergelijking geeft ons het geprojecteerde punt . Omdat de punten integer-coördinaten hebben, beschouwen we dat een punt op een lijnstuk ligt indien én als:

Maak hiervoor een functie in Speler met als definitie “**boolean** isDeelVanPad(**Punt** p)”, die teruggeeft of het meegegeven Punt **p** al dan niet ligt op (tenminste) één van de lijnstukken van de **lijnreeks**.



Figuur 3: Projectie van een punt op een lijnstuk

### Deel 2

In dit deel gaan we het grafische aspect realiseren. Je krijgt hiervoor al de klasse **TronPanel** (een JPanel).

Een speler wordt voorgesteld door een zwart puntje (tip: gebruik DrawLine() met een gelijk begin- en eindpunt) en beweegt zich in het begin naar rechts. Spelers kunnen enkel draaien met hoeken van 90°, en worden bestuurd met **twee** toetsen (Speler 1 met “A” en “Z”, Speler 2 met de linker- en rechterpijltjestoets). De speler draait respectievelijk naar links of naar rechts.

Het afgelegde pad wordt beschreven door een reeks van lijnstukken (gebruik hiervoor de **lijnreeks** van Speler uit deelvraag 1). De kleur van het pad is voor Speler 1 oranje (RGB code {255, 128, 0}) en voor Speler 2 blauw {64, 64, 255}. Wanneer een Speler een pad raakt (zowel van zichzelf als van de tegenstander) wordt hij verwijderd van het spel.

Plaats tenslotte een zwart vierkant rond het speelveld met coördinaten {16,16} (linkerbovenhoek) en {384,384} (rechteronderhoek). Als een Speler dit vierkant raakt, wordt hij ook verwijderd. Wanneer je op de spatiebalk drukt, wordt het spel herstart.



Figuur 4: Grafische voorstelling van het spel "Tron"