Aufgabe 3: Wandertag

Team-ID: 00178

Team-Name: Team-Name

Bearbeiter/-innen dieser Aufgabe:
Matthew Greiner

November 11, 2024

Contents

1 Lösungsidee / Ansatz **1**

2 Umsetzung **1**
2.1 Einlesung der Daten 1
2.2 Berechnung der drei besten Streckenlängen 2

3 Beispiele **3**

4 Quellcode **6**

1 Lösungsidee / Ansatz

In dieser Aufgabe haben wir n Personen, die jeweils eine minimale und maximale Strecke angegeben haben, in der sie einen Lauf mitrennen würden. Diese Werte müssen erst aus einer Datei gelesen werden und in einen passenden Datentyp gespeichert werden. Um drei Strecken zu erhalten, bei denen die meisten Personen mitrennen, zu erhalten, muss man zuerst eine Liste aus Strecken erstellen, die infrage kommen. Hierfür kann man alle minimale und maximale Streckenwünsche verwenden, da diese als Intervallgrenzen gesehen werden können, und so dieses Intervall alle möglichen Werte zwischen diesen Grenzen, abdeckt. Anschließend müssen für jede infragekommende Strecke die Teilnehmer ermittelt werden, die bereit wären mitzurennen. Dies kann man mit einer Hashmap implementieren, bei dem die Schlüssel die Streckenlängen sind und die Werte von einem Datentyp sind, in dem alle Teilnehmer enthalten sind, die bei der Strecke (Schlüsselwert) mitlaufen würden. Nun werden alle möglichen dreier Kombinationen an Streckenlängen erstellt und verglichen, um drei Streckenlängen zu finden, bei dem es die meisten Mitläufer gäbe. Dies kann mit drei verschachtelten for-Schleifen, die über jede Strecke iterieren, umgesetzt werden. Abschließend wird die Gesamtzahl an Teilnehmer bei den drei Strecken ausgegeben, sowie die Strecken und die Mitrennenden selbst.
Da so eine Implementierung mit drei for-Schleifen eine asymptotische Laufzeit von $O(n^3)$ hat, müssen einige Optimierungen vorgenommen werden, damit die Ausführung des Programms auch bei einer Eingabe von $n \geq 800$ (Beispielaufgabe 7) in einer realistischen Zeit ausgeführt werden kann. → Siehe Umsetzung

2 Umsetzung

2.1 Einlesung der Daten

Um die Daten aus einer Datei zu lesen, wird die Datei als ein String gespeichert, an den newline Zeichen ("\n") gespalten und jede Zeile als ein Element im Array gespeichert. Jedes Element wird an den

Leerzeichen gespalten, um an die Zahlenwerte zu kommen, die als Attribute in einem Objekt gespeichert werden. Jedes dieser "Streckenobjekte" entspricht somit einer Zeile aus der Eingabedatei. Diese Objekt werden in einer Liste gespeichert.

2.2 Berechnung der drei besten Streckenlängen

Um die drei am besten passende Streckenlängen aus den eingelesenen Streckenwünschen zu berechnen, bietet es sich an, das Problem in folgende Teilschritte zu gliedern:

1. **Erstellung einer Liste aus möglichen Streckenlängen**
   Um die minimalen und maximalen Streckenwünsche zu einer Liste hinzuzufügen, kann man über alle Wünsche iterieren und in ein Hashset hinzufügen, da Sets keine Duplikate enthalten können. Da so alle mehrfach vorkommenden Längen automatisch ausgefiltert werden, können diese Elemente anschließend in eine Liste hinzugefügt werden.

2. **Zuordnung der Strecken auf Mitrenner der Strecken**
   Um die infragekommende Strecken auf die Teilnehmer, die bei dieser Streckenlänge mitlaufen würden, zuzuordnen, bietet sich eine Hashmap an. Hierbei entsprechen die Schlüssel die Streckenlängen (Ganzzahl) und die Werte der Hashmap die mitlaufenden Teilnehmer (BitSet). Hierfür bieten sich BitSets an, da nur gespeichert werden muss, ob ein Teilnehmer mitläuft. Diese BitSets sind von der Länge n (Anzahl an Personen mit Streckenwünschen), bei dem eine 1 (true) an der Stelle x bedeutet, das der Teilnehmer x bei dieser Streckenlänge mitmachen würde. Analog zu dem, bedeutet eine 0 (false) an der Stelle y, dass der Teilnehmer y bei dieser Streckenlänge nicht mitmacht. Diese Wahl an Datenstruktur wird begründet durch die platzeffiziente Speicherung der Teilnehmer und die daduch ermöglichten schnellen Mengenoperationen (Vereinigung, Schnittmenge etc.) durch bitweise Operatoren. So kann z.B. mit der Schnittmenge - logical or Operator - schnell überprüft werden, wieviele Teilnehmer insgesamt bei unterschiedlichen Strecken teilnehmen würden. Nun kann für dieses Teilproblem eine Schleife über die möglichen Strecken und eine innere Schleife, die über die Streckenwünsche iteriert, ein BitSet für jede Strecke angelegt werden.

3. **Parallisierte Ausführung zur Suche der besten dreier Streckenkombination**
   Um für alle möglichen Kombinationen von drei Streckenlängen zu berechnen, wie viele Teilnehmer sie abdecken kann, müssen drei verschachtelte for-Schleifen verwendet werden. Diese Verschachtelung hat folgende Struktur:
   1. Äußere Schleife (i): Wählt die erste Strecke in einer Dreierkombination.
   2. Mittlere Schleife (j): Wählt die zweite Strecke, nach i.
   3. Innere Schleife (k): Wählt die dritte Strecke, nach j.

   Zusammen prüfen die Schleifen jede mögliche Kombination von drei Streckenlängen, um die beste Teilnehmeranzahl zu berechnen. Um die insgesamte Teilnehmeranzahl von drei verschiedenen Strecken zu berechnen, können auf die BitSets, die die Teilnehmender bei einer Strecke darstellen, logische or Operationen ausgeführt werden, um die Vereinigung der ganzen Teilnehmer zu bekommen. Nun müssen nur alle 1-er Bits gezählt werden, um die Gesamtzahl zu erhalten.
   Dennoch hat diese Schleifenstruktur eine asymptotische Laufzeit von $O(n^3)$ als Folge. Asymptotisch gesehen, kann dies nicht mit dieser Implementierung nicht verhindert werden. Allerdings können mehrere Threads zur Ausführung verwendet werden, um die reale Laufzeit des Programmes, je nach Anzahl der Threads, drastisch zu verringern. Man kann die Klasse IntegerStream aus Java verwenden, um parallele Streams zu erhalten, um die äußere Schleife (i) zu parallelisieren. Die Schleife wird auf alle Kerne des Prozessors verteilt, was bedeutet, dass die Berechnungen in den Iterationen parallel ablaufen. Somit könnte die Laufzeit $O(\frac{n^3}{T})$, wobei T die Anzahl der Theads ist, annähern. Asymptotisch gesehen macht die parallelisierte Implementierung keinen Unterschied, praktisch ist allerdings oft ein großer Unterschied zu spüren.
   Für diese parallelisierte Implementierung ist es wichtig, außerhalb von den einzelnen Threads das beste Ergebnis von Streckenkombinationen zu speichern. In Java können hierfür Atomic Variablen verwendet werden, um unteilbare und threadsichere Operationen zu garantieren. Daher deklariert man vor den Schleifen, Atomic Variablen, die die maximale Anzahl an Teilnehmer speichern, sowie die zugehörigen Streckenlängen und Teilnehmer. In der innersten Schleife, müssen die Zugriffe auf diese Variablen in einem "synchronized" Block stehen, damit nur ein Thread an einem Zeitpunkt auf diese Variablen zugreift.

4. **Umwandlung der Bitsets in lesbare Formen und Ausgabe**
   Nun werden die BitSets in eine Liste von Teilnehmernummer konvertiert, indem die Indizes des 1-er Bits in dem BitSet in eine Liste hinzugefügt werden. Die Indizes entsprechen den Teilnehmernummer, beginnend mit 0. Zuletzt werden die berechneten Streckenlängen, Teilnehmender und die gesamte Teilnehmeranzahl auf die Konsole in einem lesbaren Format ausgegeben.

3 Beispiele

Hier die Ausgaben des Programmes zu den Beispielen auf der BwInf-Webseite: Zusätzlich wird die Ausführungzeit in Sekunden (intel i3-12100f) ausgegeben.

1. **wandern1.txt**

```
Maximale Teilnehmerzahl: 6
Beste Streckenlängen: [22, 51, 64]
Strecke 22 m: Teilnehmer [0, 1]
Strecke 51 m: Teilnehmer [3, 4]
Strecke 64 m: Teilnehmer [5, 6]

Ausführungszeit in Sekunden: 0.022
```

2. **wandern2.txt**

```
Maximale Teilnehmerzahl: 6
Beste Streckenlängen: [10, 60, 90]
Strecke 10 m: Teilnehmer [4, 5]
Strecke 60 m: Teilnehmer [0, 2, 3]
Strecke 90 m: Teilnehmer [1]

Ausführungszeit in Sekunden: 0.021
```

3. **wandern3.txt**

```
Maximale Teilnehmerzahl: 10
Beste Streckenlängen: [22, 66, 92]
Strecke 22 m: Teilnehmer [6, 8, 9]
Strecke 66 m: Teilnehmer [0, 1, 2, 5]
Strecke 92 m: Teilnehmer [3, 4, 7]

Ausführungszeit in Sekunden: 0.022
```

4. **wandern4.txt**

```
Maximale Teilnehmerzahl: 79
Beste Streckenlängen: [524, 811, 922]
Strecke 524 m: Teilnehmer [64, 65, 3, 67, 6, 7, 71, 9, 10, 11, 12, 78, 80, 81,
                           82, 19, 83, 20, 23, 24, 25, 90, 91, 97, 34, 98, 99,
                           37, 38, 39, 40, 42, 47, 54, 56, 57, 58, 60]
Strecke 811 m: Teilnehmer [0, 64, 1, 3, 4, 5, 71, 8, 73, 74, 75, 13, 14, 16,
                           18, 82, 19, 20, 21, 87, 88, 25, 26, 27, 29, 31, 32,
                           96, 97, 34, 43, 48, 51, 53, 55, 59, 62]
Strecke 922 m: Teilnehmer [34, 66, 3, 35, 36, 69, 72, 75, 46, 48, 18, 50, 51,
                           22, 86, 89, 92, 30, 62, 94]

Ausführungszeit in Sekunden: 0.138
```

5. **wandern5.txt**

```
Maximale Teilnehmerzahl: 153
Beste Streckenlängen: [36696, 60828, 88584]
Strecke 36696 m: Teilnehmer [128, 1, 2, 5, 133, 7, 135, 136, 11, 14, 144, 17, 145,
                             20, 21, 151, 26, 27, 29, 32, 161, 163, 37, 38, 39, 40,
                             168, 42, 170, 44, 45, 174, 47, 175, 176, 177, 178, 180,
                             181, 54, 182, 186, 189, 190, 63, 191, 64, 192, 66, 194,
                             67, 195, 197, 71, 199, 72, 73, 78, 86, 88, 89, 91, 93,
                             94, 95, 96, 97, 98, 102, 103, 104, 106, 107, 108, 110,
                             111, 113, 114, 115, 117, 118, 121, 123, 124, 126, 127]
Strecke 60828 m: Teilnehmer [128, 1, 129, 130, 131, 4, 132, 5, 133, 135, 137, 10,
                             138, 11, 13, 143, 144, 145, 147, 20, 21, 149, 151, 24,
                             26, 154, 29, 157, 158, 31, 32, 33, 161, 34, 162, 37,
                             166, 39, 40, 168, 41, 42, 170, 171, 45, 173, 175, 48,
                             177, 178, 52, 54, 182, 184, 57, 186, 62, 63, 191, 64,
                             192, 65, 193, 194, 195, 196, 72, 81, 84, 85, 87, 91,
                             94, 96, 101, 102, 104, 105, 107, 108, 111, 113, 114,
                             117, 118, 124]
Strecke 88584 m: Teilnehmer [130, 3, 131, 5, 6, 8, 9, 10, 139, 140, 13, 142, 15,
                             16, 144, 147, 20, 148, 22, 150, 151, 153, 26, 29, 157,
                             30, 33, 34, 165, 39, 169, 170, 172, 175, 53, 56, 184,
                             59, 187, 191, 76, 77, 80, 84, 90, 94, 100, 105, 114,
                             116, 119, 120]
```

6. **wandern6.txt**

```
Maximale Teilnehmerzahl: 330
Beste Streckenlängen: [42834, 74810, 92920]
Strecke 42834 m: Teilnehmer [0, 257, 260, 261, 262, 9, 266, 18, 276, 277, 278, 279,
                             280, 25, 283, 284, 30, 32, 288, 33, 290, 293, 40, 41,
                             297, 43, 299, 44, 300, 45, 301, 46, 50, 306, 51, 307,
                             52, 53, 54, 311, 56, 312, 58, 314, 315, 60, 318, 319,
                             64, 324, 69, 325, 70, 71, 328, 74, 75, 81, 85, 341, 86,
                             87, 88, 344, 346, 348, 94, 350, 353, 99, 100, 102, 358,
                             360, 362, 110, 366, 367, 112, 368, 113, 114, 117, 373,
                             120, 121, 377, 379, 125, 382, 128, 384, 129, 385, 132,
                             390, 136, 137, 393, 396, 399, 144, 145, 148, 404, 149,
                             405, 150, 152, 412, 157, 414, 415, 417, 418, 163, 165,
                             166, 429, 174, 175, 431, 432, 179, 436, 181, 184, 185,
                             189, 447, 193, 194, 195, 451, 196, 197, 456, 457, 202,
                             203, 460, 205, 463, 210, 211, 469, 214, 215, 473, 221,
                             477, 478, 223, 481, 228, 487, 490, 235, 236, 237, 496,
                             241, 244, 248, 249, 251, 254, 255]
Strecke 74810 m: Teilnehmer [2, 259, 4, 7, 11, 12, 268, 19, 277, 25, 27, 284, 29,
                             287, 32, 33, 34, 35, 291, 292, 37, 39, 295, 297, 42, 298,
                             299, 301, 46, 47, 306, 51, 309, 54, 55, 311, 59, 316, 318,
                             320, 322, 67, 70, 326, 71, 329, 330, 331, 334, 79, 81,
                             83, 341, 342, 87, 343, 90, 346, 347, 93, 349, 351, 352,
                             98, 354, 102, 103, 106, 107, 108, 110, 366, 111, 367, 368,
                             114, 115, 371, 117, 374, 120, 379, 381, 383, 128, 129, 385,
                             386, 131, 132, 389, 135, 391, 140, 141, 397, 142, 398, 143,
                             145, 146, 403, 148, 404, 151, 152, 409, 412, 157, 416, 419,
                             164, 165, 428, 429, 174, 430, 175, 176, 432, 178, 179, 435,
                             438, 184, 186, 442, 443, 188, 444, 447, 449, 194, 195, 451,
                             197, 453, 198, 200, 456, 458, 206, 468, 213, 469, 214, 470,
                             215, 216, 474, 476, 221, 477, 478, 480, 225, 229, 230, 487,
                             234, 490, 494, 496, 243, 244, 249, 254]
```

```
Strecke 92920 m: Teilnehmer [258, 6, 7, 264, 11, 12, 268, 272, 275, 20, 27, 31, 32,
                             289, 295, 297, 299, 46, 47, 49, 305, 306, 63, 322, 70, 334,
                             79, 335, 336, 84, 342, 345, 346, 91, 92, 93, 349, 97, 356,
                             106, 363, 369, 370, 115, 123, 381, 126, 383, 387, 132, 388,
                             138, 394, 395, 141, 142, 143, 409, 154, 410, 157, 420, 168,
                             425, 426, 171, 177, 433, 179, 435, 180, 182, 184, 190, 448,
                             450, 195, 451, 197, 198, 454, 200, 206, 462, 465, 467, 469,
                             470, 215, 479, 224, 480, 226, 482, 484, 486, 231, 232, 238,
                             497, 247, 253]

Ausführungszeit in Sekunden: 12.843
```

7. **wandern7.txt**

```
Maximale Teilnehmerzahl: 551
Beste Streckenlängen: [39520, 76088, 91584]
Strecke 39520 m: Teilnehmer [512, 3, 4, 5, 517, 519, 8, 520, 9, 10, 11, 523, 524, 15,
                             527, 16, 528, 529, 530, 531, 532, 21, 533, 26, 27, 539, 28,
                             29, 541, 30, 545, 547, 37, 550, 40, 41, 42, 44, 45, 558,
                             559, 566, 569, 58, 60, 61, 573, 63, 576, 65, 577, 69, 581,
                             70, 582, 583, 584, 75, 76, 588, 78, 82, 594, 596, 85, 87,
                             88, 601, 90, 603, 94, 95, 98, 611, 613, 103, 615, 104, 617,
                             106, 619, 110, 622, 623, 114, 115, 627, 117, 630, 631, 633,
                             123, 635, 637, 126, 638, 640, 643, 132, 134, 136, 648, 649,
                             651, 141, 658, 147, 148, 663, 152, 665, 154, 666, 668, 157,
                             159, 671, 160, 672, 161, 162, 163, 675, 165, 678, 169, 681,
                             170, 173, 685, 174, 175, 688, 179, 180, 182, 184, 700, 190,
                             191, 704, 705, 195, 707, 708, 710, 712, 207, 209, 210, 723,
                             212, 725, 214, 215, 727, 728, 218, 730, 219, 732, 733, 224,
                             736, 739, 228, 229, 741, 742, 747, 237, 238, 751, 755, 244,
                             756, 245, 757, 760, 762, 251, 252, 254, 766, 767, 261, 773,
                             775, 265, 778, 780, 269, 782, 272, 786, 276, 789, 790, 794,
                             283, 795, 284, 797, 286, 287, 799, 291, 292, 294, 298, 301,
                             303, 304, 307, 308, 309, 311, 313, 319, 320, 321, 325, 326,
                             331, 336, 340, 342, 347, 351, 353, 356, 357, 360, 363, 365,
                             370, 372, 374, 375, 379, 380, 381, 382, 384, 386, 390, 393,
                             396, 398, 400, 403, 407, 409, 412, 414, 421, 423, 425, 433,
                             436, 441, 444, 446, 448, 449, 453, 462, 464, 466, 469, 471,
                             475, 476, 478, 485, 488, 493, 496, 498, 500, 502, 504, 505,
                             509, 511]
Strecke 76088 m: Teilnehmer [0, 6, 8, 10, 11, 12, 15, 20, 532, 22, 534, 536, 25, 537,
                             538, 27, 539, 541, 32, 544, 34, 37, 549, 550, 551, 41, 555,
                             46, 49, 561, 562, 563, 565, 54, 566, 57, 569, 58, 572, 61,
                             573, 62, 63, 64, 577, 66, 68, 69, 581, 582, 583, 584, 73,
                             586, 588, 589, 590, 79, 591, 80, 592, 594, 85, 86, 600, 91,
                             92, 604, 605, 94, 607, 609, 98, 610, 611, 100, 109, 621, 110,
                             622, 111, 624, 113, 115, 628, 119, 631, 121, 122, 634, 124,
                             636, 125, 126, 638, 128, 640, 644, 133, 645, 646, 136, 138,
                             651, 141, 145, 657, 147, 660, 663, 152, 665, 157, 671, 672,
                             677, 166, 169, 170, 174, 687, 690, 179, 691, 183, 695, 184,
                             696, 185, 186, 698, 187, 699, 188, 700, 192, 704, 194, 706,
                             195, 197, 710, 711, 712, 201, 202, 715, 204, 716, 718, 208,
                             210, 725, 727, 216, 217, 729, 219, 732, 222, 735, 224, 737,
                             228, 740, 229, 741, 231, 743, 233, 746, 237, 749, 243, 755,
                             244, 756, 757, 759, 248, 249, 762, 251, 763, 764, 765, 766,
                             255, 767, 256, 257, 258, 770, 772, 776, 267, 270, 782, 784,
                             785, 276, 277, 790, 280, 794, 284, 798, 289, 290, 295, 298,
                             300, 304, 305, 307, 311, 312, 315, 317, 319, 325, 334, 337,
```

```
                                339, 342, 343, 346, 348, 349, 352, 358, 359, 360, 366, 368,
                                370, 373, 375, 379, 383, 385, 388, 394, 395, 403, 409, 417,
                                420, 422, 424, 425, 426, 427, 428, 434, 440, 449, 454, 455,
                                456, 460, 468, 469, 472, 475, 478, 479, 480, 481, 487, 488,
                                491, 494, 496, 498, 505, 506, 507]
    Strecke 91584 m: Teilnehmer [514, 521, 11, 12, 14, 526, 19, 538, 27, 540, 541, 546, 548,
                                37, 549, 550, 551, 554, 43, 556, 557, 46, 561, 53, 565, 54,
                                566, 55, 56, 57, 569, 59, 61, 62, 574, 64, 66, 68, 580, 581,
                                582, 72, 584, 74, 587, 79, 80, 592, 81, 92, 605, 94, 606, 607,
                                610, 100, 101, 102, 620, 109, 111, 113, 626, 629, 632, 121,
                                636, 125, 130, 644, 646, 135, 647, 141, 653, 654, 657, 146,
                                150, 152, 664, 153, 674, 164, 677, 167, 169, 682, 171, 172,
                                686, 176, 695, 184, 185, 697, 698, 187, 188, 192, 704, 194,
                                195, 197, 198, 711, 713, 204, 718, 719, 213, 217, 729, 224,
                                228, 741, 231, 744, 233, 748, 237, 754, 243, 757, 758, 248,
                                249, 761, 251, 764, 765, 255, 257, 259, 776, 779, 781, 270,
                                782, 784, 275, 787, 276, 277, 790, 792, 282, 794, 314, 318,
                                334, 342, 343, 360, 362, 368, 369, 370, 375, 377, 379, 387,
                                391, 395, 397, 401, 405, 406, 411, 413, 415, 416, 417, 418,
                                419, 425, 429, 432, 439, 452, 456, 467, 469, 475, 478, 479,
                                480, 487, 488, 489, 492, 498, 505, 508, 510]
```

Ausführungszeit in Sekunden: 67.418

4 Quellcode

Einlesung der Daten

```java
/**
* Method to parse input file with distance wishes by possible participants
*
* @param input - Inputfile as String
* @return List<Wish> - List of participants' distance wishes
*/
public static List<Wish> parseInput(String input) {
    List<Wish> wishes = new ArrayList<>();
    // Save every line in Array of Strings
    String[] lines = input.split("\n");
    int totalCountWishes = Integer.parseInt(lines[0].trim());

    // For every line split at space and save as a Wish object
    for (int i = 1; i <= totalCountWishes; i++) {
        String[] parts = lines[i].trim().split("␣");
        int min = Integer.parseInt(parts[0]);
        int max = Integer.parseInt(parts[1]);
        wishes.add(new Wish(min, max));
    }
    return wishes;
}
```

Drei beste Streckenlängen berechnen

```java
/**
* Calculate 3 distances for max amount of participants
*
* @param wishes - List of distance wishes
* @return Result - 3 best distances for max amount of participants
*/
public static Result calculateBestDistances(List<Wish> wishes) {
  // 1.
  // Get all possible distances, just checking every min & max wish
  // Put into set for unique values
  Set<Integer> possibleDistancesSet = new HashSet<>();
```

```
    for (Wish wish : wishes) {
        possibleDistancesSet.add(wish.min);
        possibleDistancesSet.add(wish.max);
    }

    // Convert to list to sort
    List<Integer> possibleDistances = new ArrayList<>(possibleDistancesSet);
    Collections.sort(possibleDistances);

    // 2.
    // Create BitSets
    // Map: Key - Distance; Value - Participants (Bitset)
    Map<Integer, BitSet> participantsPerDistance = new HashMap<>();
    for (int i = 0; i < possibleDistances.size(); i++) {
        int distance = possibleDistances.get(i);
        BitSet participant = new BitSet();
        for (int j = 0; j < wishes.size(); j++) {
            Wish w = wishes.get(j);
            if (w.min <= distance && distance <= w.max) {
                // Set Bit to true if participating
                participant.set(j);
            }
        }
        participantsPerDistance.put(distance, participant);
    }

    // 3.
    // Declare and initialize atomic variables for use in parallel
    AtomicInteger maxCountParticipants = new AtomicInteger(0);
    AtomicReference<List<Integer>> bestDistances = new AtomicReference<>(null);
    AtomicReference<List<Set<Integer>>> bestParticipantDistribution = new
      AtomicReference<>(null);

    // Parallelize outer i loop with Integer Stream
    IntStream.range(0, possibleDistances.size() - 2).parallel().forEach(i -> {
        BitSet iWish = participantsPerDistance.get(possibleDistances.get(i));

        // J loop
        for (int j = i + 1; j < possibleDistances.size() - 1; j++) {
            BitSet jWish = participantsPerDistance.get(possibleDistances.get(j));

            // Create union of i and j participants
            BitSet ijUnion = (BitSet) iWish.clone();
            ijUnion.or(jWish);

            for (int k = j + 1; k < possibleDistances.size(); k++) {
                BitSet kWish = participantsPerDistance.get(possibleDistances.get(k));

                // Create union of i, j, and k participants
                BitSet participantSetUnion = (BitSet) ijUnion.clone();
                participantSetUnion.or(kWish);

                int totalParticipants = participantSetUnion.cardinality();

                // Update best result
                // synchronized since multiple threads might try to right at the same time
                synchronized (maxCountParticipants) {
                    if (totalParticipants > maxCountParticipants.get()) {
                        // Current best total participants
                        maxCountParticipants.set(totalParticipants);
                        // Current best distances
                        bestDistances.set(
                                Arrays.asList(possibleDistances.get(i),
      possibleDistances.get(j),
                                        possibleDistances.get(k)));

                        // 4.
                        // Convert BitSet to Set<Integer>
                        List<Set<Integer>> currentDistribution = Arrays.asList(
                                convertBitSetToSet(iWish),
                                convertBitSetToSet(jWish),
                                convertBitSetToSet(kWish));
                        bestParticipantDistribution.set(currentDistribution);
```

```
                    }
                }
            }
        }
    });

    return new Result(maxCountParticipants.get(), bestDistances.get(),
      bestParticipantDistribution.get());
}
```

BitSet zu einem lesbaren Format konvertieren

```
/**
* Helper method to convert bitset to human readable Integer set
*
* @param bitSet
* @return Set<Integer>
*/
private static Set<Integer> convertBitSetToSet(BitSet bitSet) {
    Set<Integer> set = new HashSet<>();
    bitSet.stream().forEach((x) -> set.add(x));
    return set;
}
```