Solution using Branch-and-bound

- Create three solution structures: parent, child1 and child2
- Start with the empty solution vector, i.e. set all the vector's items to 0. ← parent
- Compute value and upper bound of parent
- Set current best solution to parent's value
- Store parent in priority queue
- Then:

```
While ( queueSize > 0  and  parent.upperBound > currentBestSolution)
    parent = removeMax()   // from priority queue
    child1.solutionVector = parent.solutionVector
    child2.solutionVector = parent.solutionVector
    Add a 1 to child1's solutionVector
    Add a 0 to child2's solutionVector
    Compute value and upperBound for child1
    if (feasible)
        if (child1.value > currentBestSolution)
            currentBestSolution = child1.value
            finalSolution = child1.solutionVector
        insertToQueue(child1)
    Compute value and upperBound for child2
    if (feasible)
        if (child2.value > currentBestSolution)
            currentBestSolution = child2.value
            finalSolution = child2.solutionVector
        insertToQueue(child2)
return finalSolution
```

Complexity of B&B $O(2^n)$, i.c. exponential but on average way better than brute force because many paths are ignored.