Google Code Jam Cheat Sheet

<http://blog.benoitblanchon.fr/> - v1.0

# Algorithm design

## Ask yourself before coding

* Is a suboptimal solution accepted, i.e. can you use a heuristic?
* If so, what input would fool your heuristic? Is it acceptable?
* Can you divide the problem into smaller ones?
* Can you build the answer from a base case, like N=1?
* Would it be easier if the data was sorted?

## Classic algorithms and techniques

1. Brute force / Backtracking
2. Graph/Tree traversal: BFS and DFS
3. Divide and Conquer / Binary Search
4. Base case and build
5. Dynamic programming (cache and reuse data)
6. Path finding (e.g A\*, Dijkstra)
7. Heuristics (e.g greedy algorithm)

## Your implementation is too slow?

1. Are you visiting the same graph node several time?
2. Can you save the intermediate results to avoid computing the same thing again and again?
3. Can you pre-compute some useful data?
4. Can you quickly eliminate a large set of candidate solutions (pruning)?

## Maximum problem size, depending on complexity

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Complexity** |  |  |  |  |  |  |  |  |
| **N max** |  |  |  |  |  |  |  |  |

## Data type limits

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| int | Long | BigInteger | doubles | decimal |
|  |  |  | , 15 digits | , 28 digits |

# Mathematics

## Quadratic equation

|  |  |  |
| --- | --- | --- |
|  |  |  |

## Combinations or Sets (order is ignored)

|  |  |  |
| --- | --- | --- |
| *k* elements among *n* | Number of pairs | With repeated elements: |

## Permutations (order counts)

|  |  |
| --- | --- |
| *k* elements among *n* | Number of permutations |

## Probability

|  |  |  |
| --- | --- | --- |
| A and B | A or B | A knowing B (Bayes theorem) |

## Workaround the accuracy of floating point values

|  |  |  |
| --- | --- | --- |
|  |  | Don’t do strict comparisons.  Accept a small error (epsilon). |

## Prime numbers

|  |  |  |
| --- | --- | --- |
| function gcd(a, b)  while b ≠ 0  t := b  b := a mod b  a := t  return a |  | Coprime |

# Stuck?

1. Read the problem again, make sure you leverage every piece of information  
   (including the limits, the problem size and the statistics)
2. Try to solve by hand
3. Watch carefully the results and look for a pattern
4. Still nothing? Keep 30 to 60 minutes for the brute force implementation.

# Before you send your solution

1. Read the problem again, make sure there is no misunderstanding
2. Look at the statistics.   
   If some people fails (< 90% success), you should look for the pitfall.  
   If many people fails (< 70% success), you may consider skipping the problem.
3. Forget about the O(n!), it’s doomed to fail
4. Assert everything (input, output and intermediate values) if something goes wrong you’ll detect and fix the bug within the 4 minutes
5. Add samples to the input file: singular case (zero, negative…) and extreme cases
6. Remove all unused code and clean everything that stays in your program