- P1106 approximation ratio
- P1107 a 1-approximation algorithm produces an optimal solution, and an approximation algorithm with a large approximation ratio may return a solution that is much worse than optimal.
- Three problems are covered:
  1. The vertex-cover problem (35.1) - the approximation algorithm has a constant approximation ratio of 2
  2. The set-covering problem (35.3) - approximated cost is at most a logarithmic factor larger than the optimal cost. The approximation ratio grows as the input size n grows.
  3. The subset-sum problem (35.5) - a fully polynomial-time approximation scheme

- An **approximation scheme** for an optimization problem is an approximation algorithm that takes as input not only an instance of the problem, but also a value $\varepsilon > 0$ such that for any fixed ε , the scheme is a (1+ ε) - approximation algorithm.
- An approximation scheme is a **polynomial-time approximation scheme** if for any fixed $\varepsilon > 0$, the scheme runs in time polynomial in the input size n.
- An approximation scheme is a **fully polynomial-time approximation scheme** if it is an approximation scheme and its running time is polynomial in both 1/ ε and the input size n (i.e. any constant-factor decrease in ε comes with a corresponding constant-factor increase in the running time). For example, the scheme might have a running time of O((1/ ε)^2*n^3).

35.1 The vertex-cover problem

- P1108 The vertex-cover problem is to find a vertex cover of minimum size in a given undirected graph. This problem is the optimization version of an NP-complete decision problem.
- P1109 see APPROX-VERTEX-COVER
- Theorem 35.1
  APPROX-VERTEX-COVER is a polynomial-time 2-approximation algorithm.

- P1110 How to prove an approximation ratio? Instead of requiring that we know the exact size of an optimal vertex cover, we rely on finding a lower bound on the size.
- The set A of edges that line 4 of APPROX-VERTEX-COVER selects is actually a maximal matching in the graph G.
- A matching or independent edge set in a graph is a set of edges without common vertices. A maximal matching is a matching that is not a proper subset of any other matching. A maximal matching M is a matching that no other edges of the graph can be added to M. Adding one more edge to M stops it to be a matching.

35.3 The set-covering problem

- P1117 The set-covering problem is an optimization problem that models many problems that require resources to be allocated. Its corresponding decision problem generalizes the NP-complete vertex-cover problem and is therefore also NP-hard.
- In the simple greedy heuristic, as the size of the instance gets larger, the size of the approximate solution may grow, relative to the size of an optimal solution.
- The set-covering problem is to find a minimum size subset C ⊆ F whose member cover all of X (see 35.3 in P1118)
- The set of C is the number of sets it contains, rather than the number of individual elements in these sets, since every subset C that covers X must contain all |x| individual elements.
- P1119 The greedy method works by picking, at each stage, the set S that covers the greatest number of remaining elements that are uncovered.
- See P1119 for GREEDY-SET-COVER
- P1119 theorem 35.4
- P1122 Corollary 35.5

35.5 The subset-sum problem

- P1128 An instance of the subset-sum problem is a pair (S,t), where S is a set $\{x_1, x_2, \dots, x_n\}$ of positive integers and t is a positive integer that is the target value. This decision problem asks whether there exists a subset of S that adds up exactly to the target value t.
- In the optimization version of the subset-sum problem, we wish to find a subset of $\{x_1, x_2, \dots, x_n\}$ whose sum is as large as possible but not larger than t.
- See P1129 EXACT-SUBSET-SUM
- P1128 To implement this algorithm, we can use an iterative procedure that, in iteration I, computes the sum of all subsets of $\{x_1, x_2, \dots, x_i\}$, using as a starting point the sums of all subsets of $\{x_1, x_2, \dots, x_{i-1}\}$.
- P1129 The list L_i is a sorted list containing every element of P_i whose value is not more than t.
- Since the length of L_i can be as much as 2^i , EXACT-SUBSET-SUM is an exponential-time algorithm.
  Special case:
  It is polynomial-time when
  1. t is polynomial in |S|
  2. all the numbers in S are bounded by a polynomial in |S|

- Now we talk about a fully polynomial-time approximation scheme.
- P1129 We can derive a fully polynomial-time approximation scheme for the subset-sum problem by "trimming" each list L_i after it is created. The idea behind trimming is that if two values in L are close to each other, then since we want just an approximate solution, we do not need to maintain both of them explicitly.

- More precisely, we use a trimming parameter 0 < δ < 1.
- For every element y that was removed from list L, there is always an element z in L' that approximates y (i.e. z represents y):

$$\frac{y}{1+\delta} \leq z \leq y$$

- P1130 Trimming can dramatically decrease the number of elements kept while keeping a close (and slightly smaller) representative value in the list for each deleted element.

- P1130 TRIM
- In TRIM, a number is appended onto the returned list L' only if it is the first element of L or if it cannot be represented by the most recent number placed into L'.

- Note: the integers in the input S = {$x_1, x_2, \ldots, x_n$ } is in arbitrary order. It is L_i that needs to be in sorted increasing order.
- P1131 APPROX-SUBSET-SUM
  Note: $\delta = \frac{\varepsilon}{2n}$ in the code
- P1132 Theorem 35.8
  APPROX-SUBSET-SUM is a fully polynomial-time approximation scheme for the subset-sum problem

- P1132
  For every element y in P_i that is at most t, there always exists an element $z \in L_i$ such that

$$\frac{y}{(1+\frac{\varepsilon}{2n})^i} \leq z \leq y$$

  Where ε is approximation parameter 0 < ε < 1

- P1133 Since the running time of APPROX-SUBSET-SUM is polynomial in the lengths of L_i , we conclude that APPROX-SUBSET-SUM is a fully polynomial-time approximation scheme.