1. P7
   1. Binary Search
2. Acid etch bath: solve for target frequencies (*t*) for crystals.
   * 1. Right-side is always an increasing function based on *t*.
     2. Guess for *t*
        1. Check other side.
        2. Too high, low
           1. Keep searching in the middle
3. Binary search
   1. Conditions for Binary search
      1. Always increasing
      2. Always decreasing
   2. Pseudocode
      1. while(high-low > epsilon) {
         1. mid = (low + high)/2;
         2. if (f(mid) < target)
            1. low mid;
         3. else
            1. high = mid;
      2. } //May cause floating point errors
      3. for (i = 0; i < 100; i++) {
         1. mid = (low + high)/2;
         2. if (f(mid) < target)
            1. low = mid;
         3. else
            1. high = mid;
      4. } //Guarantees it will end; better for floating point error
4. Conference Question
   1. Talks
      1. T1 = 120 min
      2. T2 = 80 min
      3. T3 = 90 min
      4. …
      5. etc.
   2. Access to infinite number of rooms (*k*), but…
      1. Minimize room use.
      2. Max time (MaxT) for conference.
   3. Figure out how many rooms
   4. Process
      1. Guess: *k* = 2
         1. R1
            1. T1 = 120
            2. T4 = 30
            3. T5 = 45
            4. T7 = 35
            5. Total = 230 min
         2. R2
            1. T2 = 80
            2. T3 = 90
            3. T6 = 70
            4. Total = 240 min
         3. Result: impossible.
            1. Lowest possible number of rooms is 3.
   5. In code
      1. Binary search
         1. low = 1, high = *n* (# of talks)
         2. mid = (low + high + 1)/2;
         3. if (f(mid) > maxT) low = mid + 1;
         4. else high = mid;
      2. Breaking ties: put lower room first
5. Careful approach
   1. Up to 8 planes landing
   2. Landing windows
      1. P1 [15,35]
      2. P2[5,30]
      3. P3[25,50]
      4. P4[20,40]
      5. P5[40,60]
   3. Maximize minimum interval between any pair of landings
      1. P2 5 🡪 5
      2. P1 15 🡪 10
      3. P4 25 🡪 10
      4. P3 25 🡪 10
      5. P5 35 🡪 10
      6. P5 60 🡪 25
   4. Difficulties
      1. Unsure what order to land.
         1. Try all permutations
      2. Hard to maximize gap since there are lots of possible landing times.
      3. Unclear when to definitely land second plane.
         1. P2, p1, p4, p3, p5
         2. Can I land the planes in this order with a minimum gap of 13?
            1. P2 🡪 5
            2. P1 🡪 18

Why waste time?

Landing at 18 instead of 19 maximizes time for later

* 1. See getmax() function in approach.c
     1. Return space->start-1e-9 <= time && time<= space->end+1e-9
        1. //1e-9 epsilon fudge factor added for tolerance
     2. if (k == length return getMaxTime(perm, times, length);
        1. //And next lines