Problem Reduction

6.6

Transform and Conquer

- Algorithms based on the idea of transformation
  - Transformation stage: Problem instance is modified to be more amenable to solution
  - Conquering stage: Transformed problem is solved

Transform and Conquer

- Algorithms based on the idea of transformation
  - Transformation stage: Problem instance is modified to be more amenable to solution
  - Conquering stage: Transformed problem is solved
- Major variations are for the transform to perform:
  - Instance simplification
  - Different representation
  - Problem reduction

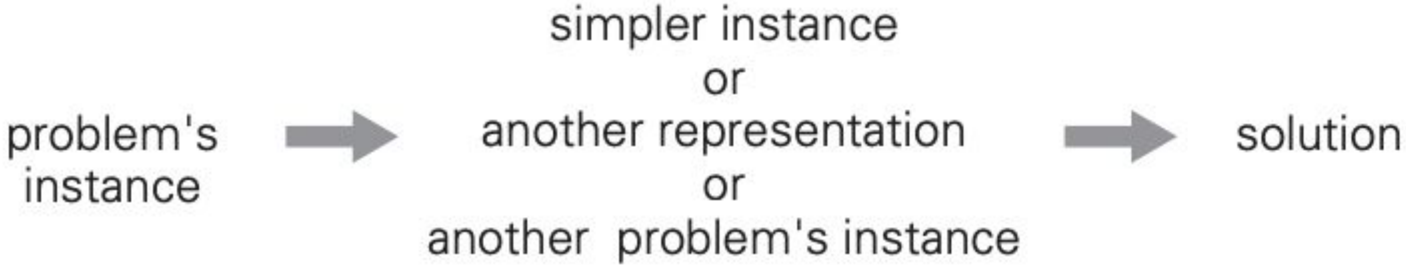
Variations of the transformation

Transformation to a simpler or more convenient instance of the same problem—we call it **instance simplification**.

Transformation to a different representation of the same instance—we call it **representation change**.

Transformation to an instance of a different problem for which an algorithm is already available—we call it **problem reduction**.

Transform and conquer strategy



Problem Reduction

If you need to solve a problem but don’t know where to start, reduce it to another problem that you know how to solve

Problem Reduction

Problem decomposition refers to the problem-solving process that computer scientists apply to solve a complex problem by breaking it down into parts that can be more easily solved.

Oftentimes, this involves modifying algorithm templates and combining multiple algorithm ideas to solve the problem at hand

Problem Reduction

However, modifying an algorithm template is just one way that we can solve a problem. An alternative approach is to represent it as a different problem, one that can be solved without modifying an algorithm.

Reduction is a problem decomposition approach where an algorithm designed for one problem can be used to solve another problem.

Problem Reduction

1. Modify the input so that it can be framed in terms of another problem.
2. Solve the modified input using a standard (unmodified) algorithm.
3. Modify the output of the standard algorithm to solve the original problem.

Let’s define problem reduction

- To solve an instance of problem A:
  - Transform the instance of problem A into an instance of problem B
  - Solve the instance of problem B
  - Transform the solution to problem B into a solution of problem A
- We say that problem A reduces to problem B

Least Common Multiple Example

Remember earlier in the semester, we worked on Greatest Common Divisor GCD

Now, let’s learn Least Common Multiple (LCM)

- Example: LCM(24, 36)

Problem Reduction to solve LCM

Solution technique: reduce LCM to GCD:

- Instance of Problem A: LCM(24, 36)
- Instance of Problem B: GCD(24, 36)

What is GCD(24, 36)?

Problem Reduction to solve LCM

Solution technique: reduce LCM to GCD:

- Instance of Problem A: LCM(24, 36)
- Instance of Problem B: GCD(24, 36)

What is GCD(24, 36)?

- Solution of B: GCD(24, 36) = 12 (by Euclid's algorithm)

Problem Reduction to solve LCM

Transform solution of B into solution of A:

- LCM(24, 36) = 24*36 / GCD(24, 36) = 24*36 / 12 = 72

$$\operatorname{lcm}(m, n) = \frac{m \cdot n}{\gcd(m, n)},$$

Problem Reduction to solve Element Uniqueness

Solve Element Uniqueness problem: given a set of two dimensional points, are all elements unique?

Solution technique:

- Reduce Element Uniqueness to Closest Pair
- Solve Closest Pair
- Transform solution of Closest Pair to solution of Uniqueness:
  - Answer to Element Uniqueness is yes if Closest Pair distance > 0

Problem Reduction to solve Element Uniqueness

A note on my last bullet point from the previous slide..

We can reduce from element uniqueness to closest pair, based on the observation that the elements of the input list are distinct if and only if the distance between the closest pair is bigger than zero.

Problem Reduction to solve Element Uniqueness

The element uniqueness problem asks, given a list of n numbers x1 , x2 , . . . , xn , find whether any two of them are equal.

We talked about an obvious and simple algorithm to solve this problem: sort the numbers, and then scan for adjacent duplicates.

Problem Reduction to solve Element Uniqueness

The element uniqueness problem asks, given a list of n numbers x1 , x2 , . . . , xn , find whether any two of them are equal.

We talked about an obvious and simple algorithm to solve this problem: sort the numbers, and then scan for adjacent duplicates.

Since we can sort in O(n log n) time, we can solve the element uniqueness problem in O(n log n) time. We also have an Ω(n log n) lower bound for sorting, but our reduction does not give us a lower bound for element uniqueness. The reduction goes the wrong way!

Inscribe this on your forehead

To prove that problem A is harder than problem B, reduce B to A.

Inscribe this on your forehead

To prove that problem A is harder than problem B, reduce B to A.

When we reduce problem B to problem A, we are claiming that A is at least as complex as B

More on problem reduction

When we reduce A to B, we are saying "If I could solve B in some model of computation, then I could solve A in that model, too" (as long as the reduction is sufficiently simple that it can be performed within the relevant model of computation).

This means that B is at least as hard as A. But A could be easier – much easier.

Problem Reduction + Performance

- If A reduces to B then
  - If A = $\Omega(f(n))$ then B = $\Omega(f(n))$
  - In other words, B can have a higher lower bound than A, but not a lower one

Problem Reduction + Performance

- If A reduces to B then
  - If A = $\Omega(f(n))$ then B = $\Omega(f(n))$
  - In other words, B can have a higher lower bound than A, but not a lower one
- For example:
  - Element Uniqueness is known to be $\Omega(n \log n)$.
  - Therefore, Closest Pair is also $\Omega(n \log n)$

Problem Reduction + Performance

More detail on Closest Pair:

- Closest Pair = $\Omega(n \log n)$ is still true if Closest Pair's lowest bound is actually higher (eg $\Omega(n^2)$)
- But the lower bound for Closest Pair can not be lower (eg it can't be $\Omega(n)$)
- Actually, the lower bound for Closest Pair can be proved using decision trees, to be $\Omega(n \log n)$, which is also the upper and tight bound.

But be careful!

This is NOT CORRECT:

- If A reduces to B and B=$\Omega(f(n))$, then A=$\Omega(f(n))$

Remember!

Remember this: If A reduces to B, then the lower bound for B can't be less than the lower bound for A

B's lower bound can be higher, but not lower.

Linear Programming

Read this section!

Linear Programming- simplex method

I’m going to walk you through an example of a problem I dealt with often in my job as a BioMathematician (fancy name, huh)

Linear Programming- simplex method

You are working on a project that includs two of the company's products, let's call them x1 and x2. The profit on the second product is twice that on the first, so x1 + 2*x2 represents the total profit.

Linear Programming- simplex method

Clearly, the profit would be highest if your company devoted its entire production capacity to making the second type of product.

In a practical situation, however, this may not be possible; a set of constraints is introduced by such factors as availability of machine time, labor, and raw materials, etc.

Linear Programming- simplex method

For example, the second type of product may require a raw material that is limited so that no more than five can be made in any batch, so $x2$ must be less than or equal to five; i.e., $x2 \leq 5$.

Let's say the first product requires another type of material limiting it to eight per batch, then $x1 \leq 8$. If $x1$ and $x2$ take equal time to make and the machine time available allows a maximum of 10 to be made in a batch, then $x1 + x2$ must be less than or equal to 10; i.e., $x1 + x2 \leq 10$.

Linear Programming- simplex method

Two other constraints are that $x1$ and $x2$ must each be greater than or equal to zero, because it is impossible to make a negative number of either; i.e., $x1 \geq 0$ and $x2 \geq 0$.

Linear Programming- simplex method

You are tasked with finding the values of x1 and x2 for which the company's profit was maximized.

Linear Programming- simplex method

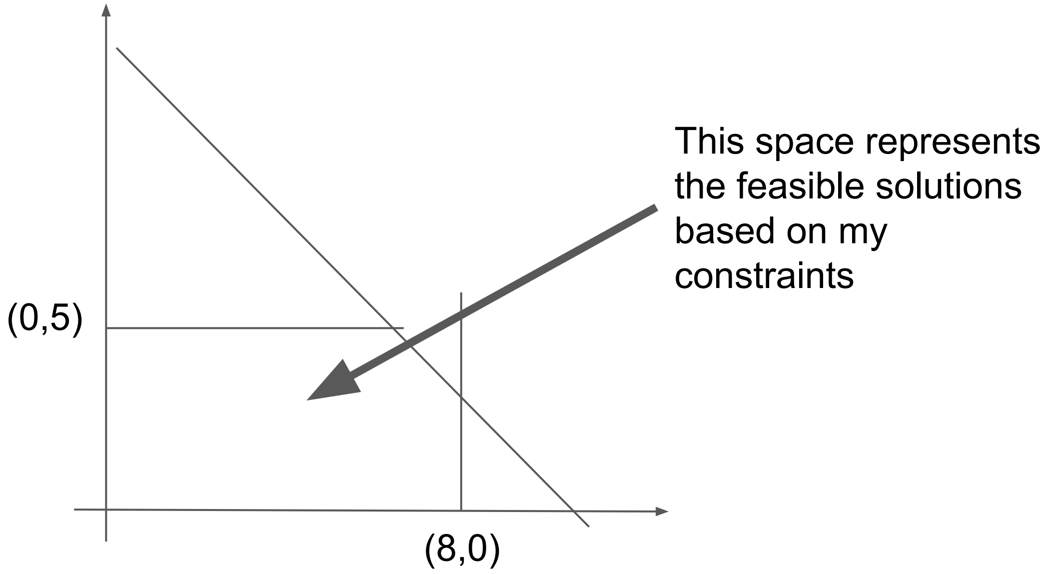
The constraint $x1 \geq 0$ means that points representing feasible solutions lie on or to the right of the x2 axis. Similarly, the constraint $x2 \geq 0$ means that they also lie on or above the x1 axis.

Application of the entire set of constraints gives the feasible solution set, which is bounded by a polygon formed by the intersection of the lines

$x1 = 0$, $x2 = 0$, $x1 = 8$, $x2 = 5$, and $x1 + x2 = 10$.

Linear Programming- simplex method

x1 = 0, x2 = 0, x1 = 8, x2 = 5, and x1 + x2 = 10.



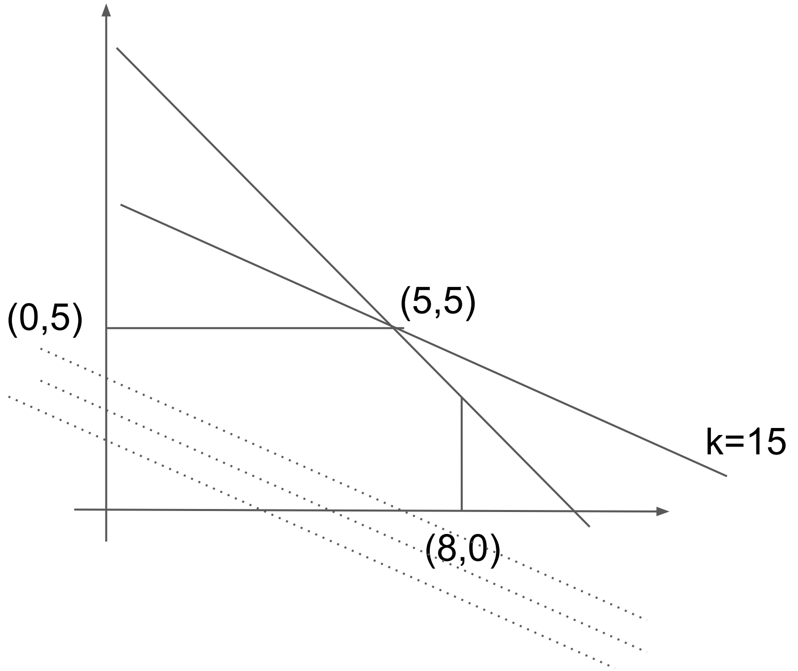
Linear Programming- simplex method

If we plot x1 + 2*x2 = k for some value of k on our graph, a family of parallel lines would be produced, and as long as they lie in the region within the constraints, they are feasible solutions.

the line for k = 15 just touches the constraint set at the point (5, 5). If k is increased further, the values of x1 and x2 will lie outside the set of feasible solutions. **Thus, the best solution is that in which equal quantities of each commodity are made.**

Linear Programming- simplex method

k = 15 just touches the constraint set at the point (5, 5)



Linear Programming- simplex method

It is no coincidence that an optimal solution occurred at a vertex, or “extreme point,” of the region.

This will always be true for linear problems, although an optimal solution may not be unique. Thus, the solution of such problems reduces to finding which extreme point (or points) yields the largest value for the objective function.

Simplex Method

Simplex method is a standard technique in linear programming for solving an optimization problem, typically one involving a function and several constraints expressed as inequalities.

The inequalities define a polygonal region, and the solution is typically at one of the vertices. The simplex method is a systematic procedure for testing the vertices as possible solutions.

Simplex Method

To understand the solution to our example problem, we need to convert the linear inequalities into equalities by introducing “slack variables”.

In linear programming, a slack variable is referred to as an additional variable that has been introduced to the optimization problem to turn a inequality constraint into an equality constraint.

A slack variable is always positive since this is a requirement for variables in the simplex method.

Simplex Method

$x_3 \geq 0$ (so that $x_1 + x_3 = 8$)

$x_4 \geq 0$ (so that $x_2 + x_4 = 5$),

$x_5 \geq 0$ (so that $x_1 + x_2 + x_5 = 10$),

and the variable x_0 for the value of the objective function (so that $x_1 + 2x_2 - x_0 = 0$).

Simplex Method

$x_3 \geq 0$ (so that $x_1 + x_3 = 8$)

$x_4 \geq 0$ (so that $x_2 + x_4 = 5$),

$x_5 \geq 0$ (so that $x_1 + x_2 + x_5 = 10$),

and the variable x_0 for the value of the objective function (so that $x_1 + 2x_2 - x_0 = 0$).

Goal: to find the largest possible x_0 satisfying the equations.

Simplex Method

One solution is to set the objective variables $x1 = x2 = 0$, which corresponds to the extreme point at the origin.

If one of the objective variables is increased from zero while the other one is fixed at zero, the objective value $x0$ will increase as desired

Simplex Method

The variable x2 produces the largest increase of x0 per unit change; so it is used first. Its increase is limited by the nonnegativity requirement on the variables. In particular, if x2 is increased beyond 5, x4 becomes negative.

Simplex Method

At $x2 = 5$, this situation produces a new solution—$(x0, x1, x2, x3, x4, x5) = (10, 0, 5, 8, 0, 5)$—that corresponds to the extreme point $(0, 5)$ in the figure. The system of equations is put into an equivalent form by solving for the nonzero variables $x0$, $x2$, $x3$, $x5$ in terms of those variables now at zero; i.e., $x1$ and $x4$. Thus, the new objective function is $x1 - 2x4 = -10$, while the constraints are $x1 + x3 = 8$, $x2 + x4 = 5$, and $x1 - x4 + x5 = 5$.

Simplex Method

It is now apparent that an increase of x_1 while holding x_4 equal to zero will produce a further increase in x_0. The nonnegativity restriction on x_3 prevents x_1 from going beyond 5. The new solution—$(x_0, x_1, x_2, x_3, x_4, x_5) = (15, 5, 5, 3, 0, 0)$—corresponds to the extreme point (5, 5) in the figure.

Simplex Method

Finally, since solving for $x0$ in terms of the variables $x4$ and $x5$ (which are currently at zero value) yields $x0 = 15 - x4 - x5$, it can be seen that any further change in these slack variables will decrease the objective value. Hence, an optimal solution exists at the extreme point (5, 5).

Simplex Method

Okay so the proof with slack variables may have been a bit confusing. Write down the equations and work through the math in the slides to see it for yourself. :)

Homework problems!

Please complete exercises 6.6 #1, 11