Summary

These notes contain two examples of primals and duals, as well as two more example reductions, as well as hints for homework problem 1c.

Duality

We will look at the formal definition of taking the dual of a linear program, and then show an easy example. The original problem is called the *primal* problem, and is stated as follows:

Maximize:

$$\sum_{j=1}^{n} c_j x_j \tag{1}$$

Subject to:

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i \qquad (i = 1, 2, \ldots, m) \tag{2}$$

$$x_j \geq 0 \qquad (j = 1, 2, \ldots, n) \tag{3}$$

The *dual* of the above *primal* (given in standard form) is defined as:

Minimize:

$$\sum_{i=1}^{m} b_i y_i \tag{4}$$

Subject to:

$$\sum_{i=1}^{m} a_{ij} y_i \geq c_j \qquad (j = 1, 2, \ldots, n) \tag{5}$$

$$y_i \geq 0 \qquad (i = 1, 2, \ldots, m) \tag{6}$$

Consider the following linear programming problem:

Maximize:

$$z = 2x_1 + 3x_2 \tag{7}$$

Subject to:

$$3x_1 + 2x_2 \leq 2 \tag{8}$$

$$-x_1 + 2x_2 \leq 5 \tag{9}$$

$$4x_1 + x_2 \leq 1 \tag{10}$$

$$x_1, x_2 \geq 0 \tag{11}$$

We can more clearly see how to take the dual by expressing this program in its matrix form:

Maximize:

$$z = \begin{bmatrix} 2 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{12}$$

Subject to:

$$\begin{bmatrix} 3 & 2 \\ -1 & 2 \\ 4 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq \begin{bmatrix} 2 \\ 5 \\ 1 \end{bmatrix} \tag{13}$$

$$x_1, x_2 \geq 0 \tag{14}$$

Then we can see that the dual problem becomes:
Minimize:

$$z' = \begin{bmatrix} 2 & 5 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \tag{15}$$

Subject to:

$$\begin{bmatrix} 3 & -1 & 4 \\ 2 & 2 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \geq \begin{bmatrix} 2 \\ 3 \end{bmatrix} \tag{16}$$

$$y_1, y_2, y_3 \geq 0 \tag{17}$$

Now we will look at another example of a linear program and its corresponding dual, but this will be a "real life" example and a "real life" dual. A hospital menu consists of two foods A and B, each of which contains certain amounts of fats, carbohydrates and proteins:

Constituent	A	B
Fat	1	4
Carbohydrate	3	2
Protein	5	3

It is necessary that each hospital meal provide at least 12 units of fat, 15 units of carbohydrates and 22 units of protein. Each ounce of food A costs 30 cents and each ounce of food B costs 25 cents, and obviously the hospital wants to minimize its expenses and yet still provide all of its patients with the appropriate amount of nutrition. Thus, this gives rise to the following linear program:

Minimize:

$$z = 30x_A + 20x_B \tag{18}$$

Subject to:

$$x_A + 4x_B \geq 12 \tag{19}$$

$$3x_A + 2x_B \geq 15 \tag{20}$$

$$5x_A + 3x_B \geq 22 \tag{21}$$

$$x_A, x_B \geq 0 \tag{22}$$

Now we will think of ourselves as directors of a firm selling artificial foods F, C and P (obviously indicating Fats, Carhybohydrates and Proteins), where one unit of F provides 1 unit of fat, etc. As directors of this firm, we want to provide the same nutritional value as foods A and B and yet not exceed the costs of those foods. We also want to **maximize** our profits. Thus, based on these constraints, we can formulate a linear program *which just happens to be the dual of the above primal*:

Maximize:

$$z' = 12y_F + 15y_C + 22y_P \tag{23}$$

Subject to:

$$y_F + 3y_C + 5y_P \leq 30 \tag{24}$$
$$4y_F + 2y_C + 3y_P \leq 25 \tag{25}$$
$$y_F, y_C, y_P \geq 0 \tag{26}$$

Now we see the canonical economic relationship between duals and primals: minimizing expenses for a buyer is the dual of maximizing profits for a seller. This example is taken from *Linear Programming* by Calvert and Voxman.

Bonnie and Clyde

Bonnie and Clyde have just robbed a bank. They have a bag of money and want to divide it up. For each of the following scenarios, either give a polynomial-time algorithm, or prove that the problem is NP-Complete. The input in each case is a list of the n items in the bag, along with the value of each.

a. There are n checks, which are, in an amazing coincidence, made out to "Bonnie or Clyde". They wish to divide the checks so that they each get the exact same amount of money.

Solution: NP-Complete

- NP: This problem is clearly in NP. The witness algorithm consists of the bag of checks, and an assignment of each of the checks to Bonnie or Clyde. Then simply sum Bonnie's and Clyde's checks and verify that the sum is the same. This is clearly polynomial time.
- NP-Hard Even though Set-Partition is a trivial reduction, we will reduce from Subset-Sum just to be difficult. An instance of Subset-Sum is a set of integers X and a bound B, where in a *yes* instance, X contains a subset of integers that sums to exactly B. Thus, to create an instance of Bonnie-and-Clyce Checks (BNCC), we let $T = \sum i = 1^n x_i$, and let each integer x_i correspond to a check of value x_i, and then we add one more check x_{n+1} such that:
  - If $B \geq T/2$, then choose a c such that $T + c = 2B$, and let $x_{n+1} = c = 2B - T$.
  - If $B < T/2$, then choose a c such that $B + c = (T + c)/2$, and let $x_{n+1} = c = T - 2B$.

  This is clearly a polynomial time conversion algorithm since we are adding only one variable and performing only a few trivial arithmetic operations. Thus, the proof:

  - $\Longrightarrow$ Assume X is a *yes* instance of *Subset-Sum*. Thus, there exists a subset of X that sums to exactly B.
    - If $B \geq T/2$, then all of the integers that sum to B are on one side of the partition and c is on the other, giving $T + c - B = T + 2B - T - B = B$, and the remaining integers in the set plus c will sum to B, and thus there exists a partition.
    - If $B < T/2$, then c is on the same side of the partition as all of the other integers that sum to B, and $B + c = B + T - 2B = T - B$, and thus a partition exists.
  - $\Longleftarrow$ Assume BNCC is a *yes* instance. Then there exists a partition, and by definition, c will either be on one side or the other.
    - If $c = 2B - T$, then $T + c = T + 2B - T = 2B$, and both sides of the partition must sum to B. Thus, the other side of the partition will consist only of integers in X and will sum to B.
    - If $c = T - 2B$, then $T + c = T + T - 2B = 2(T - B)$, and each side of the partition must sum to $(T - B)$. Thus, $c + B = T - 2B + B = T - B$, and the other elements on the c side of the partition are all elements from the original X and sum to B.

Thus, a *yes* instance of SUBSET-SUM maps to a *yes* instance of BNCC, and vice versa. Thus, since BNCC $\in$ NP and BNCC $\in$ NP-HARD, then this implies that BNCC $\in$ NP-COMPLETE.

b. There are n checks as above, but this time Bonnie and Clyde are willing to accept a split in which the difference is no larger than \$100.

Solution: NP-COMPLETE This time we do the trivial reduction from SET-PARTITION, only when we convert to the corresponding Bonnie and Clyde problem, we multiply x_i by 1000. Thus, if the algorithm produces a solution that differs by only \$100, then the two solutions must be equal (because if we divide by 1000, we get the difference in the original problem).