CSE 101- Winter ‘18
Discussion Section
Week 8

Topics for today

Reductions

Max Flow and LP

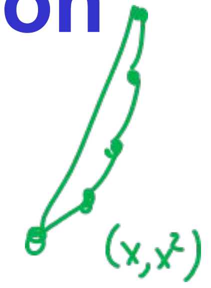
- Number Puzzle
- Circulation problem
- Maximum bipartite matching
- Bob diet plan and pill salesman

USB Problem from PA3

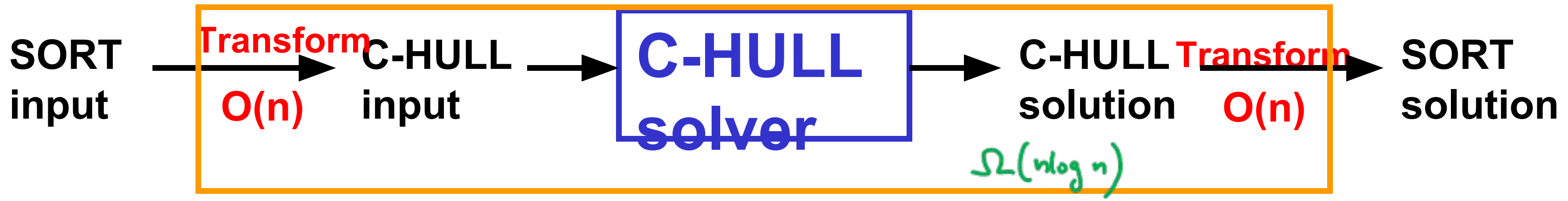
Reduction

Recall From Lecture 4 … "Reduction"

- Given: Sorting LB is $\Omega(n \log n)$
- Reduction from Sorting to Convex Hull
  - Input: an arbitrary instance I_{SORT} of SORT
  - Transform $I_{SORT} = \{x_1, x_2, \ldots, x_n\}$ into an instance
  - $I_{C\text{-}HULL} = \{(x_1,x_1^2), (x_2,x_2^2), \ldots, (x_n,x_n^2)\}$ of C-HULL
  - Solve the C-HULL problem for instance $I_{C\text{-}HULL}$
  - Transform solution of $I_{C\text{-}HULL}$ to solution of I_{SORT}



$x_i \rightarrow (x_i, x_i^2)$



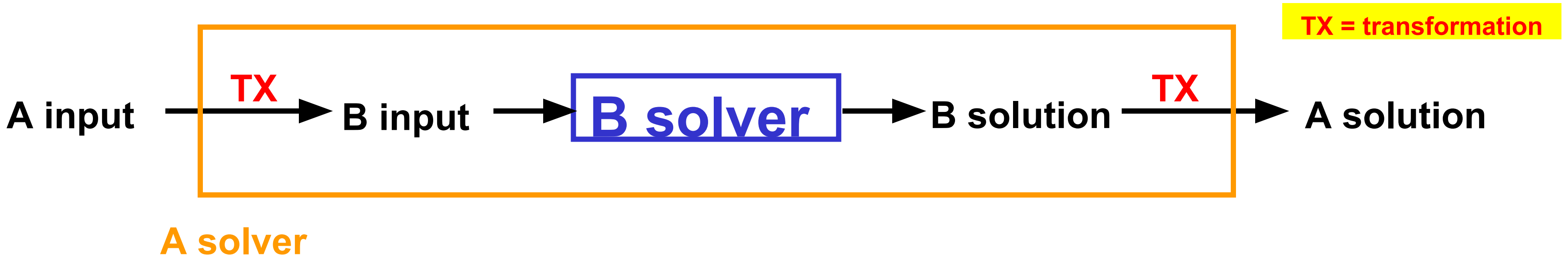
- Note: Each Transform has O(n) complexity

$\rightarrow$ C-HULL solver cannot be faster than $\Omega(n \log n)$

Reduction From A to B

- A: Problem with "known difficulty"
- B: Problem with "unknown difficulty"

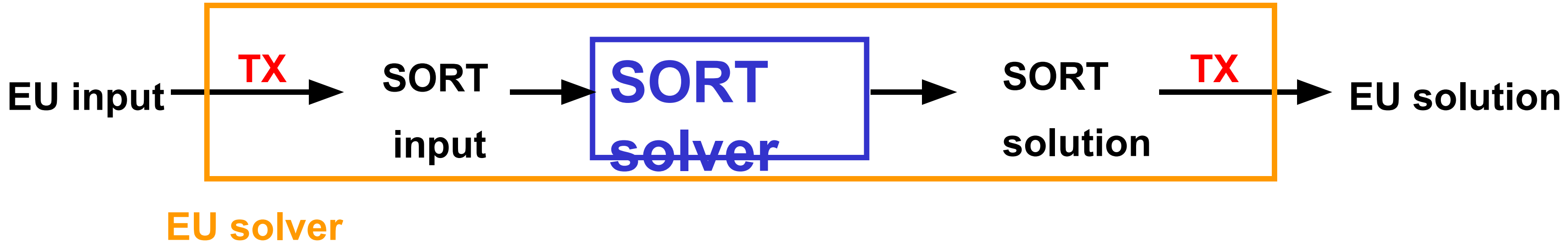
- Reduction: "Transfers the known difficulty of Problem A to Problem B"



Reduction From EU to SORT

- EU: "Element Uniqueness" – given a list of n elements, are they all distinct? == a decision problem
- SORT: Given a list of n elements, arrange them in non-decreasing order

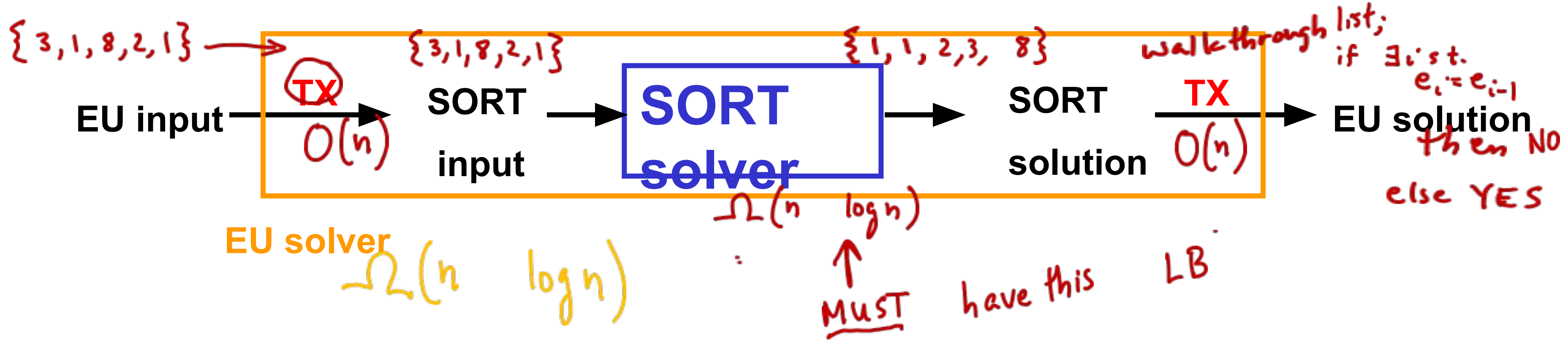
- Suppose that we are given that EU is Ω(n log n). Can we prove an Ω(n log n) LB on SORT?



Reduction From EU to SORT

- **EU:** **"Element Uniqueness" – given a list of n elements, are they all distinct?**
- **SORT:** **Given a list of n elements, arrange them in non-decreasing order**

- **Suppose that we are given that EU is Ω(n log n). Can we prove an Ω(n log n) LB on Sorting?**



Network flows

Number puzzle

You are given the sums for each row and column of an *n x n* matrix of integers in the range *1,2,..,M*. You wish to construct a matrix that is consistent with the column and row sums.

Given input *M*, row sums r_1, r_2, …. r_n, and column sums c_1,c_2, ...c_n, you should output a matrix of numbers A[i][j]. If no matrix exists, you should output “None”. Give an efficient algorithm for this problem

Number puzzle

What do we know from the problem statement?

1. Each element participates in both row sum and column sum.

$$\sum_i A[i][j] = c_j$$

$$\sum_j A[i][j] = r_i$$

$$\sum_i c_i = \sum_i r_i$$

2. Each element A[i][j] is positive and as well as in the range 1...M
3. We need any one solution (or none), not all possible solutions.

(**STOP!** think what additional information do you know before you proceed to next slide)

Number puzzle

What do we *additionally* know?

4. The range of numbers gives us constraints on the row and column sums. If any of them is less than ***n***, we know no solution is possible.

5. Column sums and row sums are not independent as each matrix element belongs to both sums.

Or in other words,

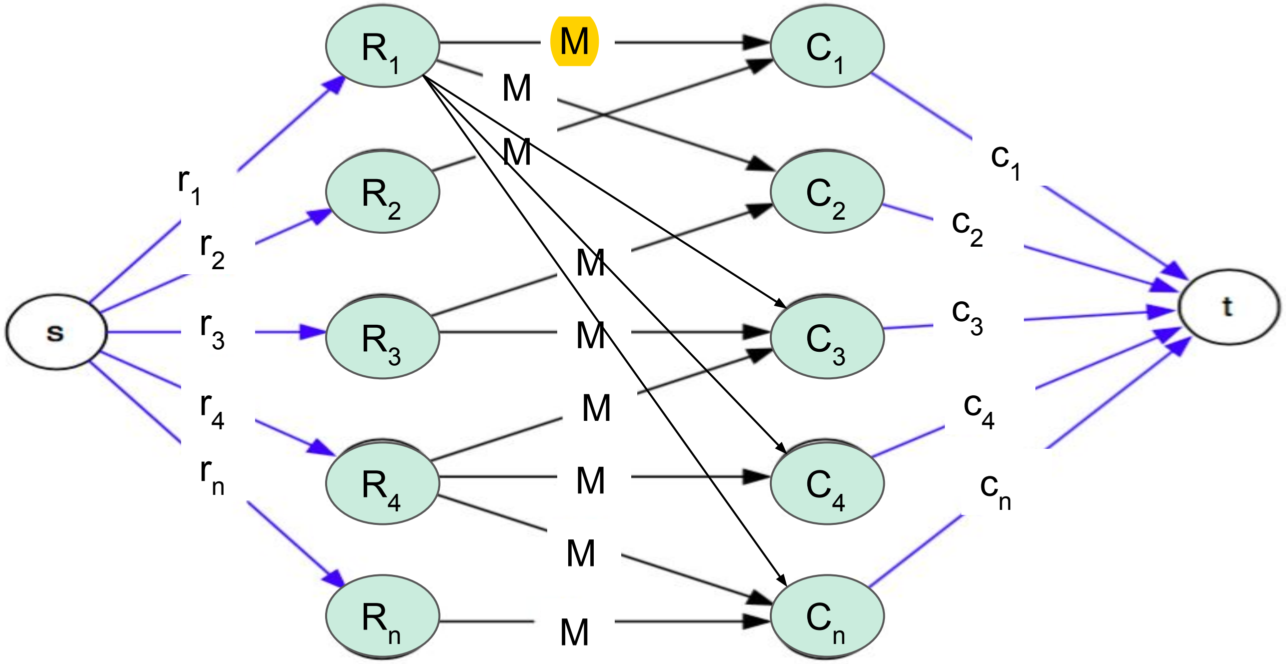
Could we distribute the sum of each row among the n columns, while respecting the column sums and other constraints?

(**STOP**! Think about what the flow network should look like? What are the nodes and the edge capacities?

Number puzzle

Objective: Interlink row and column sums. Connect source node *s* to row sums with capacity r_i, and the column sums to sink *t* with capacity c_j. Connect every row sum to every column sum with capacity M. Edge between r_i and c_j represents A[i][j]

STOP! Think about what this flow network produces and why it is incorrect!



Number puzzle

Range of values for each A[i][j] is 1...M. But our flow network can produce zero flows on some edges, which implies A[i][j] at that edge is 0.

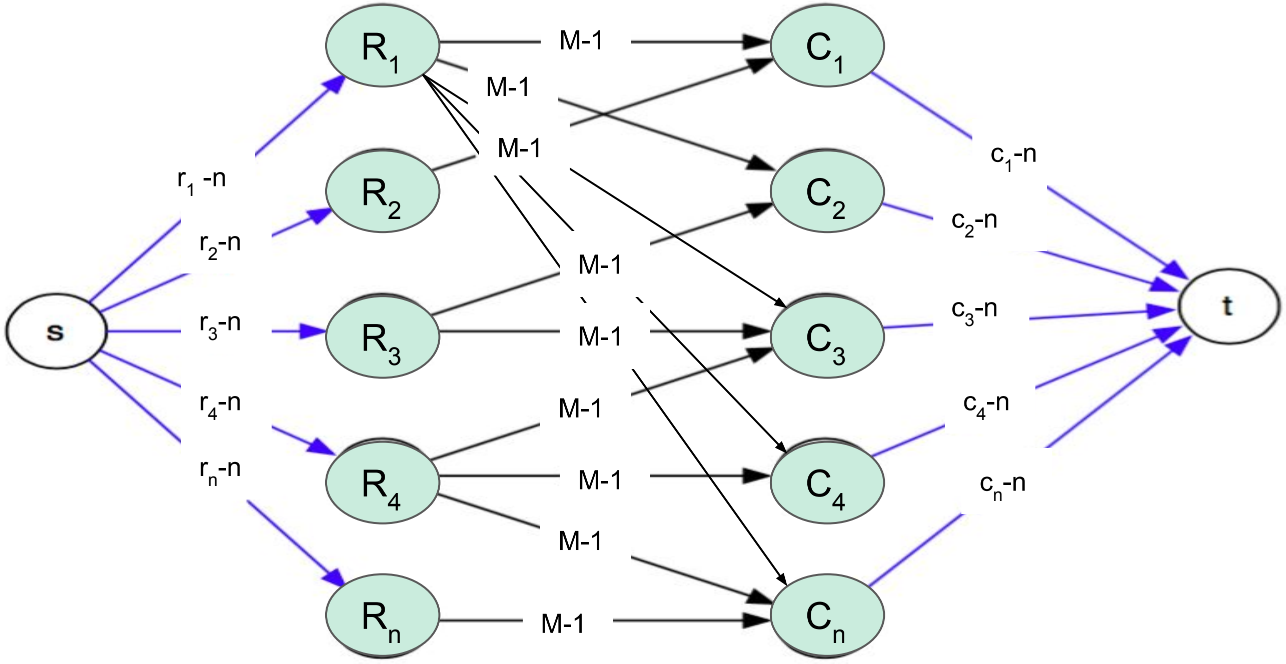
How do you fix this?

Transform the problem to generate numbers from 0...M-1 and then send maximum flow possible. When you return the matrix of elements, add +1 to the flows sent.

Number puzzle

Final flow network is given below. Say any edge between r_i and c_j has flow f(i,j). Then, A[i][j] = f(i,j) + 1 .

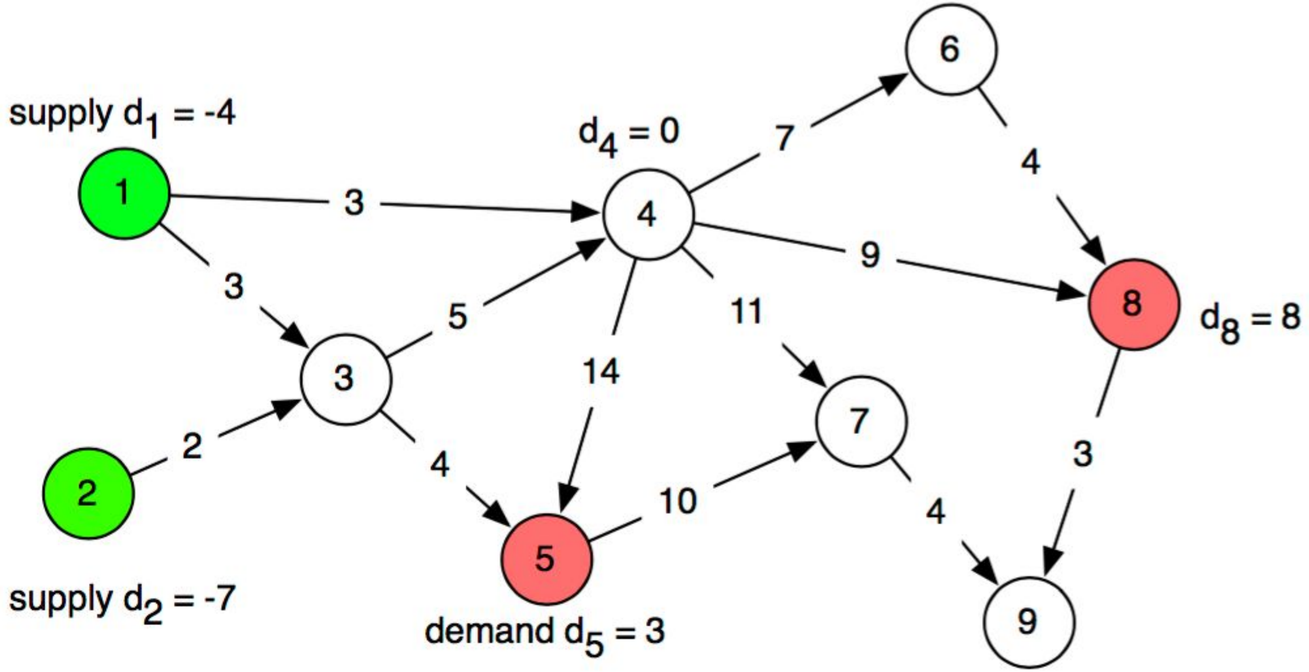
Also Max flow should be equal to $\sum_i (r_i - n)$ Else output "None" (**Why?**)



Circulation problem

- Suppose we have multiple source and multiple sinks.
- Each sink wants can absorb a certain amount of flow.
- Each source can generate certain amount of flow to give.
- **Why circulation? flow from source to destination also flows back from destination to source !**

Example



supply $d_1 = -4$
supply $d_2 = -7$
$d_4 = 0$
demand $d_5 = 3$
$d_8 = 8$
1
2
3
4
5
6
7
8
9
3
3
2
5
4
14
11
7
9
4
10
4
3

What is the goal?

- Capacity constraint : For each $e \in E$, $0 \leq f(e) \leq c_e$
- Flow Constraint : For each $v \in V$,

$$f^{in}(v) - f^{out}(v) = d_v$$

- d_v is the excess flow that should come into node.
- Negative d_v indicates the excess flow that should come out of node.
- We want to satisfy all the constraints! Feasibility problem!

Sources and Sinks?

- Let S be the set of nodes with excess flow that should come out of node.
- Let T be the set of nodes with that should come into node.

- In order for there to be a feasible flow, we must have (excess flow coming out of node= excess flow coming into node):

$$\Sigma_{s\in \mathrm{S}}\,(-d_s) = \Sigma_{t\in \mathrm{T}}\, d_t$$

Let D = $\Sigma_{t\in \mathrm{T}}\, d_t$

Reduction

- How to turn circulation problem into maximum flow problem?

Reduction

- How to turn circulation problem into maximum flow problem?
  - Add a new source s^* with an edge (s^*, s) from s^* to every node $s \in S$. (Supplies sources with flow)
  - Add a new sink t^* with an edge (t^*, t) from t^* to every node $t \in T$. (Siphons flow out of sinks)
  - Edge Capacities?

Reduction

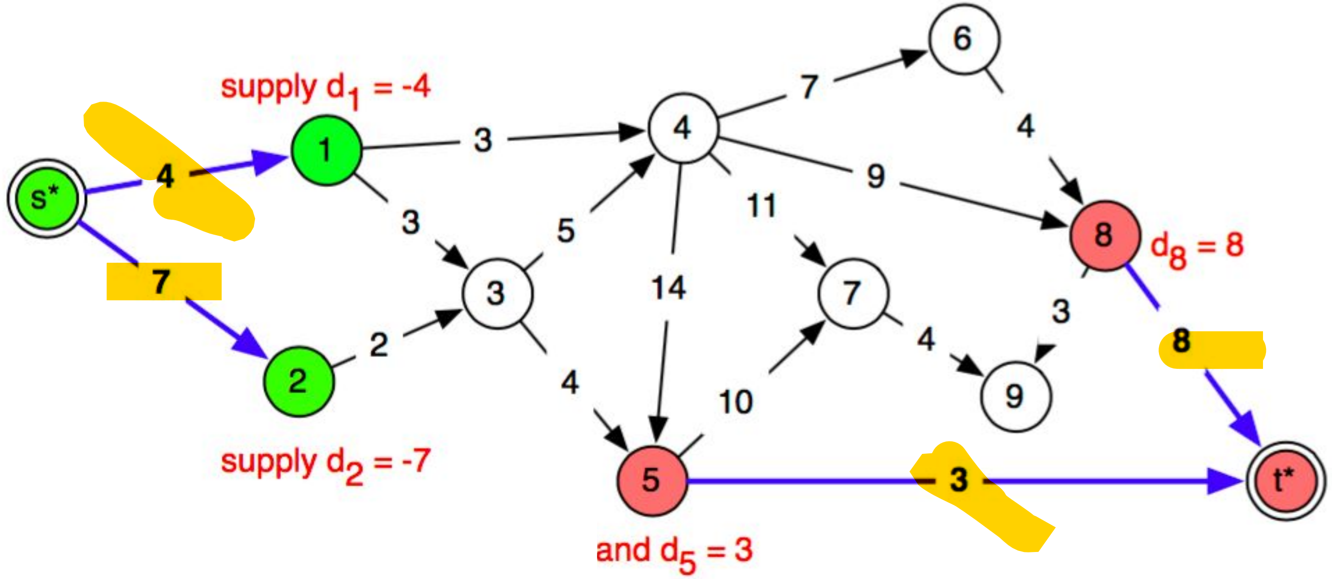
- How to turn circulation problem into maximum flow problem?
  - Add a new source s^* with an edge (s^*, s) from s^* to every node $s \in S$. (Supplies sources with flow)
  - Add a new sink t^* with an edge (t^*, t) from t^* to every node $t \in T$. (Siphons flow out of sinks)

  - Edge Capacities? Flow that has to be supplied or siphoned.

Reduction

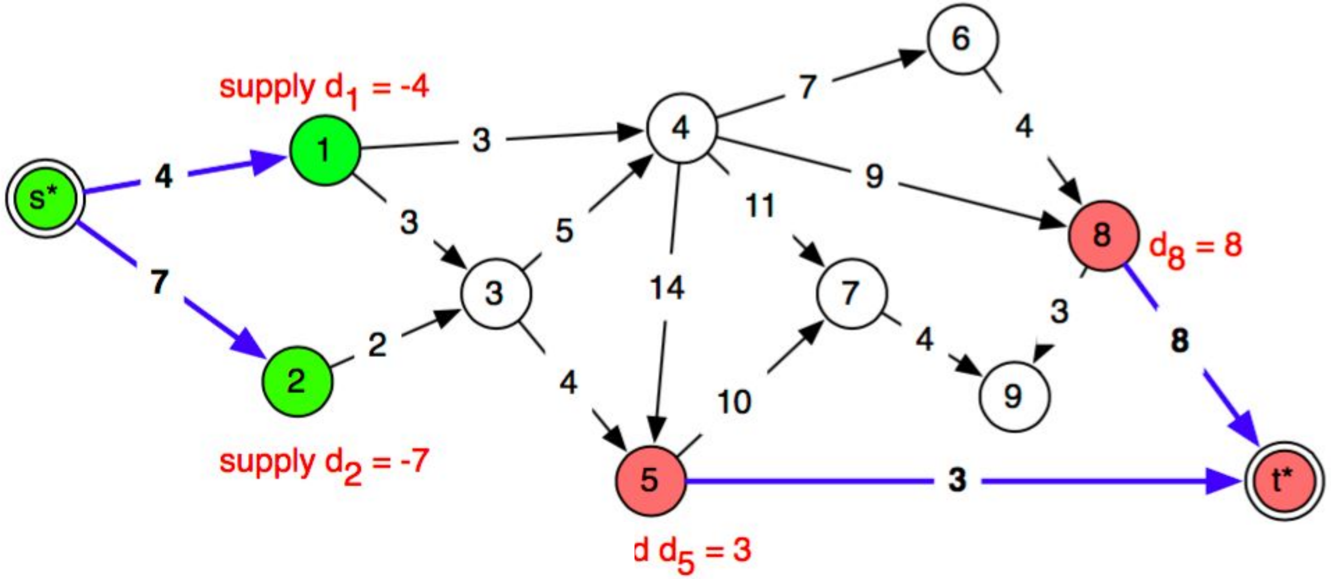
- How to turn circulation problem into maximum flow problem?
  - Add a new source s^* with an edge (s^*, s) from s^* to every node $s \in S$.
  - Add a new sink t^* with an edge (t^*, t) from t^* to every node $t \in T$.
  - The capacity of edges $(s^*, s) = -d_s$ (+ve)
  - The capacity of edges $(t^*, t) = d_t$

Reduction Example



For any sink node with no demand mentioned, its demand is 0, and therefore no edge needs to be added to t*

Reduction Example



Feasible circulation if and only if there is a flow of value $D = \sum_{t \in T} d_t$

Intuition

- Capacity of edges (s^*, s) limit supply or source node s
- Capacity of edges (t^*, t) require that d_t flow reaches each t.

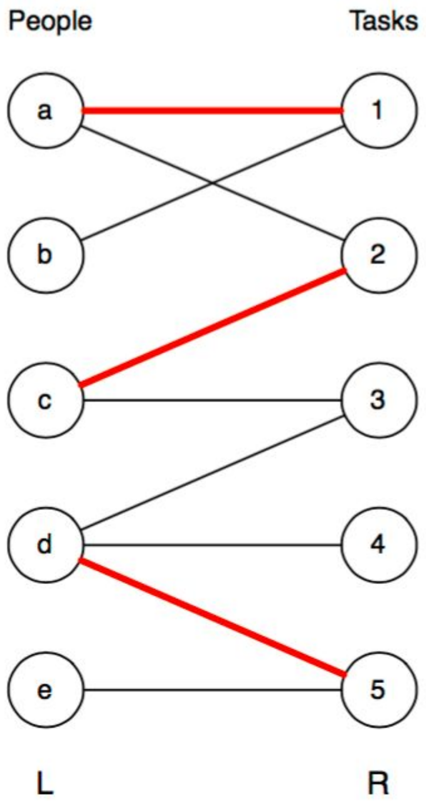
Max flow can be used to find circulation!

Maximum bipartite matching

- A matching gives an assignment of people to tasks.

- Want to get as many tasks done as possible.

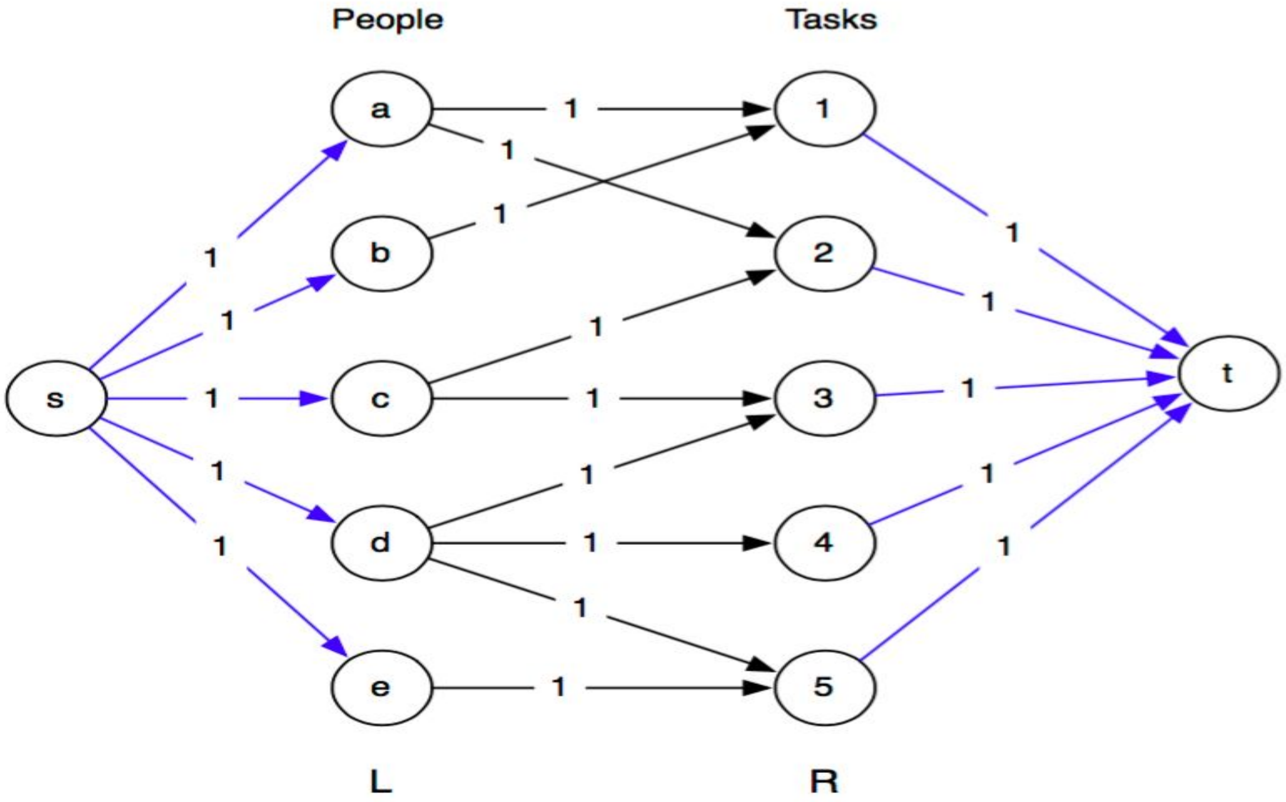
- So, want a maximum matching: one that contains as many edges as possible.



Reduction

- Given an instance of bipartite matching,
- Create an instance of network flow.
- Where the solution to the network flow problem can easily be used to find the solution to the bipartite matching.

Reduction



Reduction

- Given bipartite graph $G = (A \cup B, E)$, direct the edges from A to B.
- Add new vertices s and t.
- Add an edge from s to every vertex in A.
- Add an edge from every vertex in B to t.
- Make all the capacities 1.
- Solve maximum network flow problem on this new graph G'.

The edges used in the maximum network flow will correspond to the largest possible matching!

LP formulation

Maximize Flow

$$x_{sa} + x_{sb} + x_{sc} + x_{sd} + x_{se}$$

Subject to constraints

$$\left.\begin{aligned}
x_{sa} &= x_{a1} + x_{a2} \\
x_{sb} &= x_{b1} \\
x_{sc} &= x_{c2} + x_{c3} \\
x_{sd} &= x_{d3} + x_{d4} + x_{d5} \\
x_{se} &= x_{e5} \\
x_{a1} + x_{b1} &= x_{1t} \\
x_{a2} + x_{c2} &= x_{2t} \\
x_{c3} + x_{d3} &= x_{3t} \\
x_{d4} &= x_{4t} \\
x_{d5} + x_{e5} &= x_{5t}
\end{aligned}\right\} \text{Flow conservation}$$

all $x_{ij} \geq 0$ - Non negativity

all $x_{ij} \leq 1$ - Capacity constraint

LP formulation

Maximize Flow

$$x_{sa} + x_{sb} + x_{sc} + x_{sd} + x_{se}$$

Subject to constraints

$$x_{sa} = x_{a1} + x_{a2}$$
$$x_{sb} = x_{b1}$$
$$x_{sc} = x_{c2} + x_{c3}$$
$$x_{sd} = x_{d3} + x_{d4} + x_{d5}$$
$$x_{se} = x_{e5}$$
$$x_{a1} + x_{b1} = x_{1t}$$
$$x_{a2} + x_{c2} = x_{2t}$$
$$x_{c3} + x_{d3} = x_{3t}$$
$$x_{d4} = x_{4t}$$
$$x_{d5} + x_{e5} = x_{5t}$$
all $x_{ij} \geq 0$

all $x_{ij} \leq 1$

Solution :

Max flow = 5

$x_{sa} = 1, x_{a2} = 1, x_{2t} = 1$

$x_{sb} = 1, x_{b1} = 1, x_{1t} = 1$

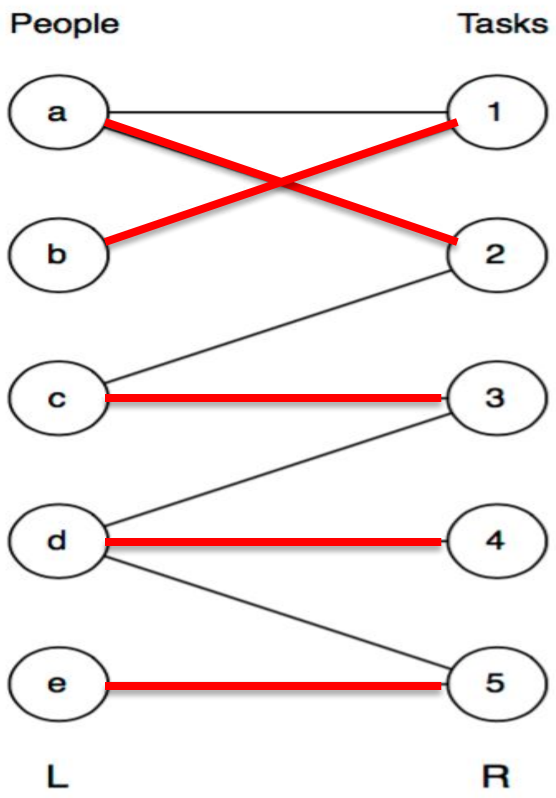
$x_{sc} = 1, x_{c3} = 1, x_{3t} = 1$

$x_{sd} = 1, x_{d4} = 1, x_{4t} = 1$

$x_{se} = 1, x_{e5} = 1, x_{5t} = 1$

All other $x_{ij} = 0$

LP formulation



Solution :

Max flow = 5

$x_{sa} = 1, x_{a2} = 1, \mathrm{x}_{2\mathrm{t}} = 1$

$x_{sb} = 1, x_{b1} = 1, \mathrm{x}_{1\mathrm{t}} = 1$

$x_{sc} = 1, x_{c3} = 1, \mathrm{x}_{3\mathrm{t}} = 1$

$x_{sd} = 1, x_{d4} = 1, \mathrm{x}_{4\mathrm{t}} = 1$

$x_{se} = 1, x_{e5} = 1, \mathrm{x}_{5\mathrm{t}} = 1$

All other $x_{ij} = 0$

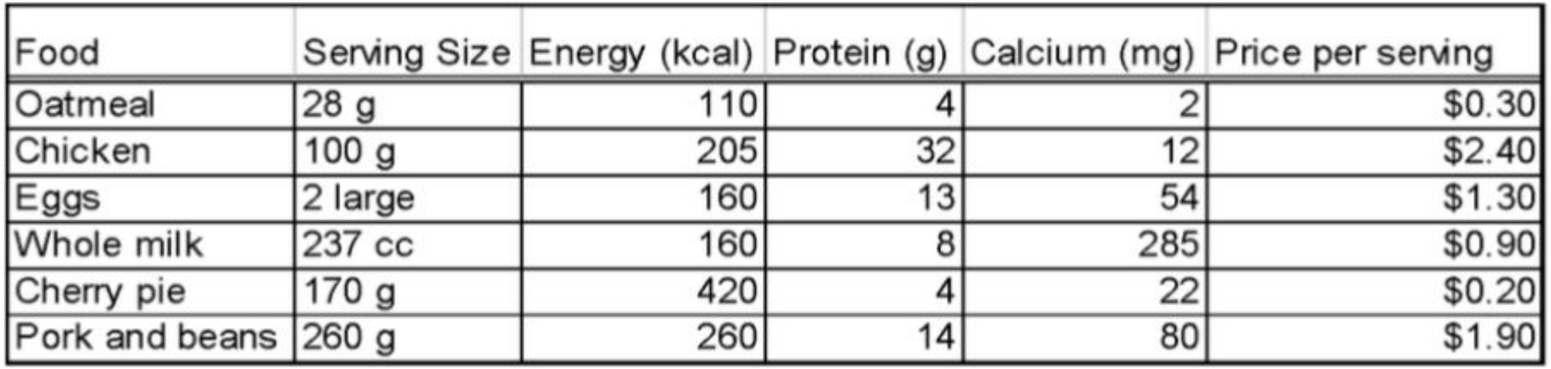
Diet Problem

Bob wants to plan a nutritious diet, but he is on a limited budget, so he wants to spend as little money as possible. His nutritional requirements are as follows:

1. 2000 kcal
2. 55 g protein
3. 800 mg calcium

Diet Problem

Bob is considering the following foods:



Food	Serving Size	Energy (kcal)	Protein (g)	Calcium (mg)	Price per serving
Oatmeal	28 g	110	4	2	$0.30
Chicken	100 g	205	32	12	$2.40
Eggs	2 large	160	13	54	$1.30
Whole milk	237 cc	160	8	285	$0.90
Cherry pie	170 g	420	4	22	$0.20
Pork and beans	260 g	260	14	80	$1.90

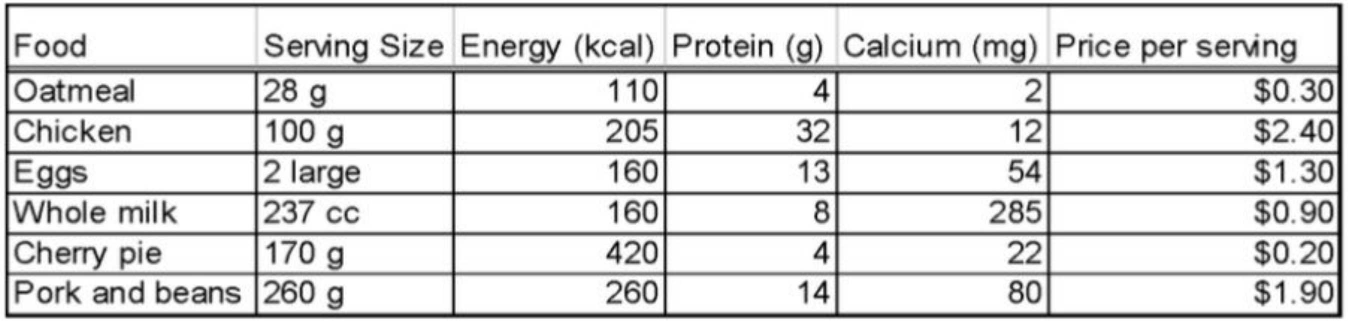
Diet Problem

We can represent the number of servings of each type of food in the diet by the variables:

x_1	servings of oatmeal
x_2	servings of chicken
x_3	servings of eggs
x_4	servings of milk
x_5	servings of cherry pie
x_6	servings of pork and beans

Diet Problem

We can represent the number of servings of each type of food in the diet by the variables:



Food	Serving Size	Energy (kcal)	Protein (g)	Calcium (mg)	Price per serving
Oatmeal	28 g	110	4	2	$0.30
Chicken	100 g	205	32	12	$2.40
Eggs	2 large	160	13	54	$1.30
Whole milk	237 cc	160	8	285	$0.90
Cherry pie	170 g	420	4	22	$0.20
Pork and beans	260 g	260	14	80	$1.90

KCAL constraint :

$$110\,x_1 + 205\,x_2 + 160\,x_3 + 160\,x_4 + 420\,x_5 + 260\,x_6 \geq 2000$$

LP Formulation

Minimize **Cost**

$$0.3x_1 + 2.40x_2 + 1.30x_3 + 0.90x_4 + 2.0x_5 + 1.9x_6$$

subject to: **Nutritional requirements**

$$\begin{aligned}
110x_1 + 205x_2 + 160x_3 + 160x_4 + 420x_5 + 260x_6 &\geq 2000 \\
4x_1 + 32x_2 + 13x_3 + 8x_4 + 4x_5 + 14x_6 &\geq 55 \\
2x_1 + 12x_2 + 54x_3 + 285x_4 + 22x_5 + 80x_6 &\geq 800
\end{aligned}$$

Bounds

$$x_1, x_2, x_3, x_4, x_5, x_6 \geq 0$$

Diet Problem

When we solve the LP we get a solution value of $6.71, which is achieved with the following menu

14.24 servings of oatmeal

0 servings of chicken

0 servings of eggs

2.71 servings of milk

0 servings of cherry pie

0 servings of pork and beans

The pill salesman

A pill salesman offers Bob energy, protein, and calcium pills to fulfill his nutritional needs. We will represent the costs of each of the pills as follows:

y_1	cost of 1 kcal energy pill
y_2	cost of 1 g protein pill
y_3	cost of 1 mg calcium pill

The pill salesman

How do I guarantee I won’t make a bad deal

Minimize **Cost**

$$0.3x_1 + 2.40x_2 + 1.30x_3 + 0.90x_4 + 2.0x_5 + 1.9x_6$$

Nutritional requirements

$$110x_1 + 205x_2 + 160x_3 + 160x_4 + 420x_5 + 260x_6 \geq 2000 \quad \mathbf{y_1 \text{ kcal}}$$

$$4x_1 + 32x_2 + 13x_3 + 8x_4 + 4x_5 + 14x_6 \geq 55 \quad \mathbf{y_2 \text{ protein}}$$

$$2x_1 + 12x_2 + 54x_3 + 285x_4 + 22x_5 + 80x_6 \geq 800 \quad \mathbf{y_3 \text{ calcium}}$$

x_1 = servings of oatmeal: The cost of the nutrients in one serving of oatmeal shouldn’t exceed the cost of just buying one serving of oatmeal:

$$110y_1 + 4y_2 + 2y_3 \leq 0.3 \quad (4\,y_2 = \text{cost of protein in oatmeal})$$

The pill salesman

The pill salesman wants to make as much money as possible, given Bob’s constraints. He knows Bob wants 2000 kcal, 55g protein, and 800 mg calcium, so his problem is as follows:

$$
\begin{array}{ll}
\text{Maximize} & 2000y_1 + 55y_2 + 800y_3 \\
\text{Subject to} & 110y_1 + 4y_2 + 2y_3 \leq 0.3 \\
 & 205y_1 + 32y_2 + 12y_3 \leq 2.4 \\
 & 160y_1 + 13y_2 + 54y_3 \leq 1.3 \\
 & 160y_1 + 8y_2 + 285y_3 \leq 0.9 \\
 & 420y_1 + 4y_2 + 22y_3 \leq 2.0 \\
 & 260y_1 + 14y_2 + 80y_3 \leq 1.9 \\
 & y_1, y_2, y_3 \geq 0
\end{array}
$$

The pill salesman

Solving the previous LP gives a total cost of $6.71 for the following pill prices :

$0.27 for 1 kcal energy pill

$0.00 for 1 g protein pill

$0.16 for 1 mg calcium pill

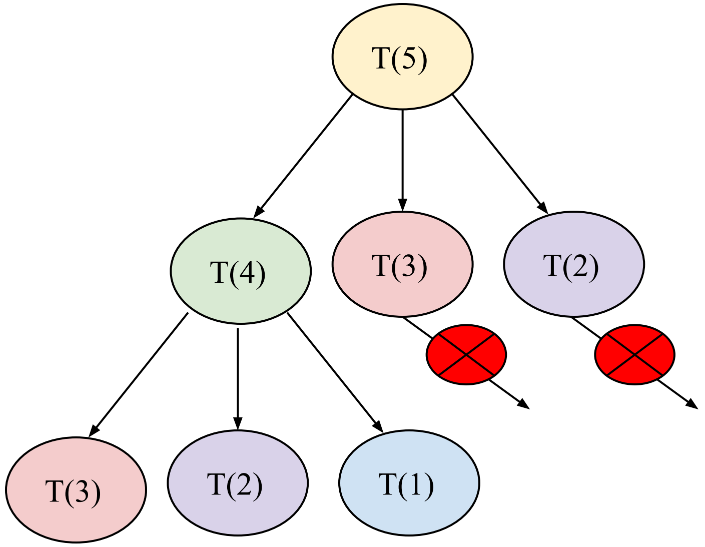
THE SAME AS THE LOWEST COST DIET!

Supplementary:USB

USB: Memoization

- Memoization is **top-down**
- For the USB problem, the algorithm starts with the total size of the USB drive, and recursively calculates the optimal number of files for smaller and smaller USB drives all the way down to the base case, memoizing the results for any subproblems so we do not have to recurse multiple times on the same subproblem

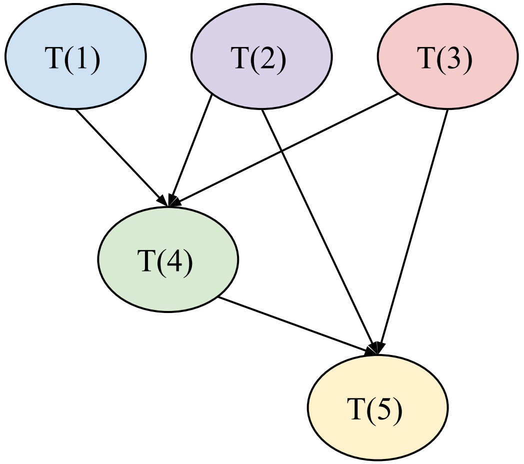
- **Example: USB(5, [1,2,3])**
- We recurse top-down, solving T(5), then T(4), then T(3), T(2), and T(1). When we encounter the problems T(3) and T(2) again in our first layer, we have saved these solutions and no longer have to recurse down their subtrees.



USB: Tabulation

- Tabulation is **bottom-up**
- Instead of starting with the total size of the USB drive, *n*, we begin calculations by calculating the optimal number of files for a USB of size *k* = 1. Then, we calculate the optimal number of files for USB[*k*+1] until we reach USB[*k* = *n*], which is our solution.

- **Example: USB(5, [1,2,3])**
- We iterate from the bottom up, solving T(1), T(2), and T(3), then using these to solve T(4), and then T(5). We do not have to recurse since we know exactly what subproblems each new subproblem will depend on.


T(1)
T(2)
T(3)
T(4)
T(5)