EC504 ALGORITHMS AND DATA STRUCTURES FALL 2020 MONDAY & WEDNESDAY 2:30 PM - 4:15 PM

g	6	6	t

Prof: David Castañón, dac@bu.edu

GTF: Mert Toslali, toslali@bu.edu

Haoyang Wang: haoyangw@bu.edu

Christopher Liao: cliao25@bu.edu

Sequence Alignment

- How similar are two sequences of symbols?
  - ❑Example: ocurrance and occurrence

o	c	u	r	r	a	n	c	e	–
o	c	c	u	r	r	e	n	c	e

6 mismatches, 1 gap

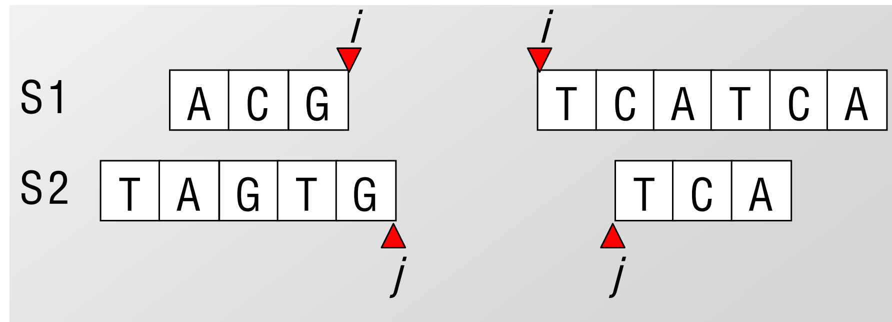
o	c	–	u	r	r	a	n	c	e
o	c	c	u	r	r	e	n	c	e

1 mismatch, 1 gap

- Applications: Bioinformatics, spell correction, machine translation, speech recognition, information extraction

Edit Distance

- Concept due to Levenshtein 1966, Needleman–Wunsch 1970
- Scoring function
  - ❑Cost of mutation (mismatch)
    - s(x,y) is cost of matching $x \neq y$
  - ❑Cost of insertion/deletion
    - δ is cost of matching x to a gap, or matching y to a gap
  - ❑Reward of correct match
    - s(x, y) is value of correctly matching when x = y



- Complex search problem
  - ❑For sequences of length 100, number of possible matches is $9 \cdot 10^{58}$

Dynamic Programming (Needleman-Wunsch) \

- Let OPT(i, j) = minimum cost of aligning prefix strings $x_1x_2\cdots x_i,\ y_1y_2\cdots y_j$
- Goal. Is to compute OPT(m,n)
- Idea: Assume we know OPT(i-1, j-1), OPT(i,j-1), and OPT(i-1,j-1):
  - ❑Case 1. OPT(i, j) matches $x_i \to y_j$: $Opt(i,j) = s(x_i, y_j) + OPT(i-1,j-1)$
  - ❑Case 2a. OPT(i, j) leaves x_i unmatched: $Opt(i,j) = \delta + OPT(i-1,j)$
  - ❑Case 2b. OPT(i, j) leaves y_j unmatched: $Opt(i,j) = \delta + OPT(i,j-1)$
- Initially, $Opt(i,0) = i\delta; \quad Opt(0,j) = j\delta$
- Iteration:

$$Opt(i,j) = \min\left\{s(x_i, y_j) + OPT(i-1,j-1), \delta + OPT(i-1,j), \delta + OPT(i,j-1)\right\}$$

Ptr(i,j) = {diag, up, left} corresponding to which term is minimized

Small Example

\

OPT(i,j) with $\delta = 2;\ s(x_i, y_j) = \begin{cases} 2, & x_i \neq y_j \\ -1, & x_i = y_j \end{cases}$

OPT =

	-	A	G	C
-	0	2	4	6
A	2	-1	1	3
A	4	1	1	3
A	6	3	3	3
C	8	5	5	2

PTR =

	-	A	G	C
-	0	Left	Left	Left
A	Up	Diag	Left	Left
A	Up	Diag	Diag	Diag
A	Up	Diag	Diag	Diag
C	Up	Up	Up	Diag

Example

Mismatch = -1
Match = 2

j		0	1	2	3	4	5	
i			c	a	d	b	d	←T
0		0	-1	-2	-3	-4	-5	
1	a	-1						
2	c	-2						
3	b	-3						
4	c	-4						
5	d	-5						
6	b	-6						
	↑ S							

c -	Score(c,-) = -1

Mismatch = -1
Match = 2

Example

	j	0	1	2	3	4	5
i			c	a	d	b	d
0		0	-1	-2	-3	-4	-5
1	a	-1	-1	1	0	-1	-2
2	c	-2	1	0	0	-1	-2
3	b	-3	0	0	-1	2	1
4	c	-4	-1	-1	-1	1	1
5	d	-5	-2	-2	1	0	3
6	b	-6	-3	-3	0	3	$\boxed{2}$

←T

↑
S

Optimal Match: Backtrack Pointers

	j	0	1	2	3	4	5	
i			c	a	d	b	d	← T
0		0	-1	-2	-3	-4	-5	
1	a	-1	-1	1	0	-1	-2	
2	c	-2	1	0	0	-1	-2	
3	b	-3	0	0	-1	2	1	
4	c	-4	-1	-1	-1	1	1	
5	d	-5	-2	-2	1	0	3	
6	b	-6	-3	-3	0	3	2	

↑
S

A Larger Example

\

$\delta = 2;$

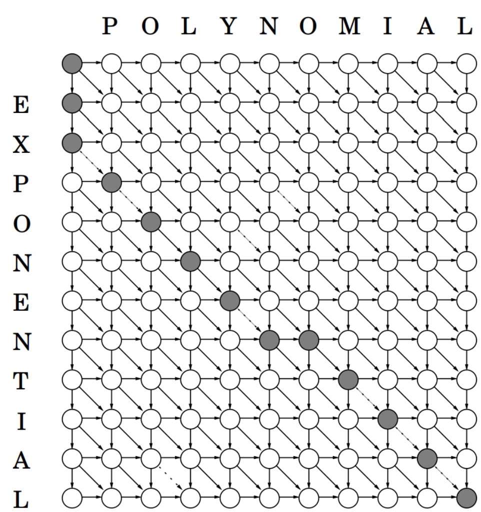
$$s(x_i, y_j) = \begin{cases} 2, & x_i \neq y_j \\ -1, & x_i = y_j \end{cases}$$

		S	I	M	I	L	A	R	I	T	Y
	0	2	4	6	8	10	12	14	16	18	20
I	2	4	1	3	2	4	6	8	7	9	11
D	4	6	3	3	4	4	6	8	9	9	11
E	6	8	5	5	6	6	6	8	10	11	11
N	8	10	7	7	8	8	8	8	10	12	13
T	10	12	9	9	9	10	10	10	10	9	11
I	12	14	8	10	8	10	12	12	9	11	11
T	14	16	10	10	10	10	12	14	11	8	11
Y	16	18	12	12	12	12	12	14	13	10	7

Another Example

$\delta = 1;$

$$s(x_i, y_j) = \begin{cases} 1, & x_i \neq y_j \\ 0, & x_i = y_j \end{cases}$$



		P	O	L	Y	N	O	M	I	A	L
	0	1	2	3	4	5	6	7	8	9	10
E	1	1	2	3	4	5	6	7	8	9	10
X	2	2	2	3	4	5	6	7	8	9	10
P	3	2	3	3	4	5	6	7	8	9	10
O	4	3	2	3	4	5	5	6	7	8	9
N	5	4	3	3	4	4	5	6	7	8	9
E	6	5	4	4	4	5	5	6	7	8	9
N	7	6	5	5	5	4	5	6	7	8	9
T	8	7	6	6	6	5	5	6	7	8	9
I	9	8	7	7	7	6	6	6	6	7	8
A	10	9	8	8	8	7	7	7	7	6	7
L	11	10	9	8	9	8	8	8	8	7	6

Sequence Matching Complexity

\

- Need to complete table of m by n
  - Length of x: m, length of y: n
- Computational complexity O(mn)
  - O(1) operations to compute new element
  - Polynomial!
- Still, may be too slow for long DNA sequences
  - 50,000 genes…
- Search for faster approximate algorithms

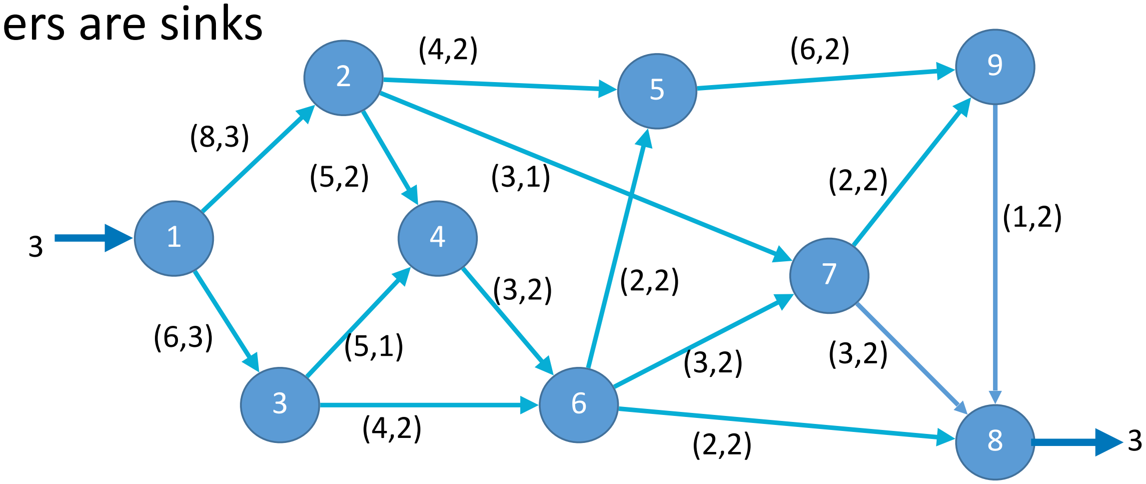
Upcoming Seminar

\

- The Second-Price Knapsack Problem: Near-Optimal Real Time Bidding in Internet Advertisement
- Abstract: In many online advertisement (ad) exchanges ad slots are each sold via a separate second-price auction. This work considers the bidder's problem of maximizing the value of ads they purchase in these auctions subject to budget constraints. This 'second-price knapsack' problem presents challenges when devising a bidding strategy because of the uncertain resource consumption: bidders win if they bid the highest amount but pay the second-highest bid unknown a priori. This is in contrast to the traditional online knapsack problem where posted prices are revealed when ads arrive and for which there exists a rich literature of primal and dual algorithms. The main results of this paper establish general methods for adapting these primal and dual online knapsack selection algorithms to the second-price knapsack problem where the prices are revealed only after bidding.
- In particular a methodology is provided for converting deterministic and randomized knapsack selection algorithms into second-price knapsack bidding strategies that purchase ads through an equivalent set of criteria and thereby achieve the same competitive guarantees.

Directed, Weighted, Capacitated Graphs

- A weighted, capacitated, directed graph is a directed graph $G = (V, E)$ along with a capacity function c(e) and a weight function w(e)
  - Capacities are positive: $c : E \rightarrow \Re^+$, weights can be real numbers: $w : E \rightarrow \Re$
- Capacity represents maximum number of simultaneous units that can use an edge, weight is cost per unit of using edge
- Some vertices are sources, others are sinks
- Min-cost flow: find minimum weight flow that matches supply with demand



Special Case: The Assignment Problem

- In many business situations, management needs to assign
  - personnel to jobs,
  - jobs to machines,
  - machines to job locations,
  - salespersons to territories
- Consider the situation of assigning n jobs to n machines.
- When a job i (=1,2,....,n) is assigned to machine j (=1,2, ...n) that incurs a cost C_{ij}
- The objective is to assign the jobs to machines at the least possible total cost

Assignment Problem Statement

\

- N persons, N objects
- Cost of matching person i to object j: $C_{ij},\ (i,j) \in E$

Variable: $x_{ij} = \begin{cases} 0 & \text{person i not assigned to object j} \\ 1 & \text{person i assigned to object j} \end{cases}$

Problem: $$\min_{\{x_{ij} \in \{0,1\}\}} \sum_{(i,j) \in E} C_{ij} x_{ij}$$

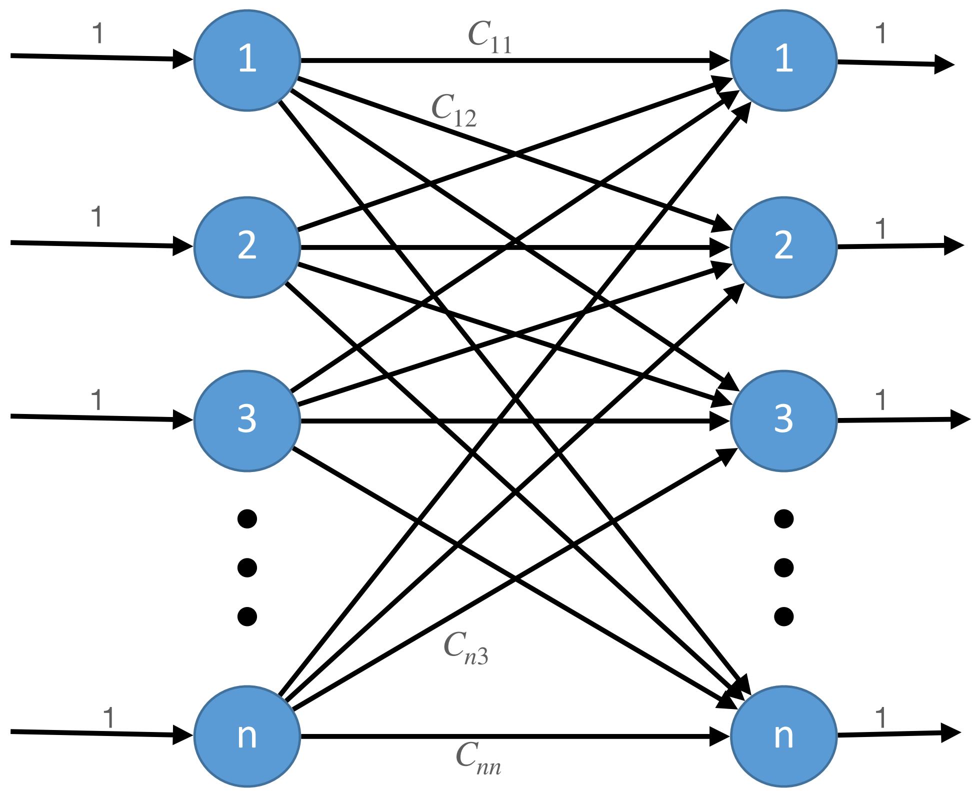
Subject to constraints:

$$\sum_{i:(i,j) \in E} x_{ij} = 1 \text{ for all } j \in 1, \ldots, n; \quad \sum_{j:(i,j) \in E} x_{ij} = 1 \text{ for all } i \in 1, \ldots, n$$

Graph Representation of Assignment

\

- Every edge has capacity 1; graph can be sparse



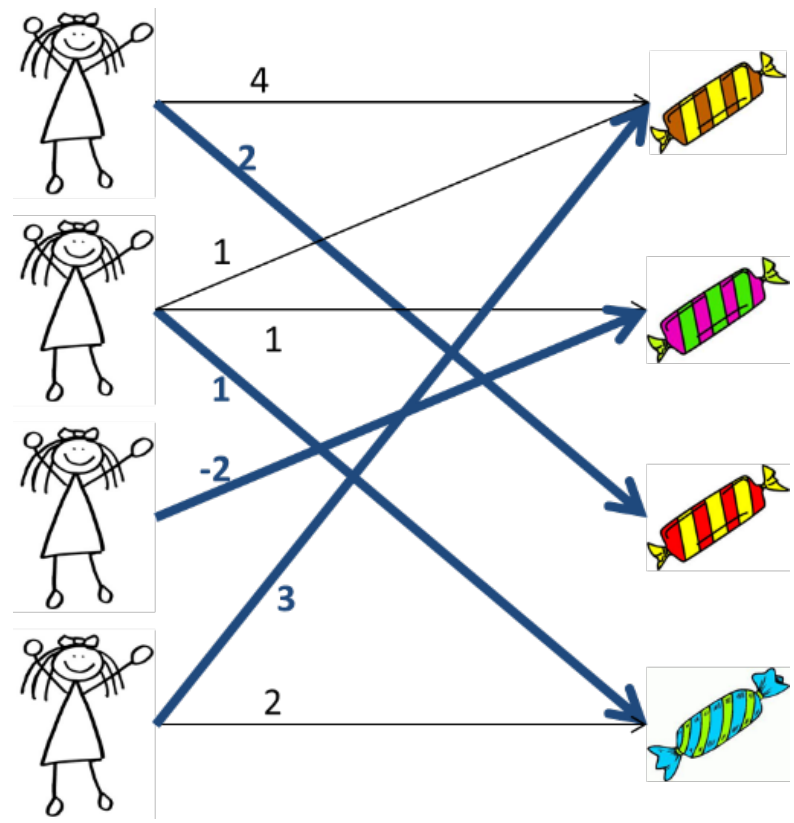
Can Formulate as Maximization

- Benefit of assignment: a_{ij}

$$\max_{\{x_{ij} \in \{0,1\}\}} \sum_{(i,j)\in E} a_{ij} x_{ij}$$

$$\sum_{i:(i,j)\in E} x_{ij} = 1 \text{ for all } j \in 1,\ldots,n$$

$$\sum_{j:(i,j)\in E} x_{ij} = 1 \text{ for all } i \in 1,\ldots,n$$



Assignment Problem Structure

\

- Conservation of flow constraints, plus integer capacities leads to special structure: Unimodularity
- Implies that optimization over flows that can be real numbers results in optimal flows that are integer
  - Can assign percentages of persons to objects, and optimal assignments are integers

Equivalent problem: Problem: $$\min_{\{x_{ij}\in[0,1]\}} \sum_{(i,j)\in E} C_{ij} x_{ij}$$

Subject to constraints:

$$\sum_{i:(i,j)\in E} x_{ij} = 1 \text{ for all } j \in 1,\ldots,n; \quad \sum_{j:(i,j)\in E} x_{ij} = 1 \text{ for all } i \in 1,\ldots,n$$

Example Application

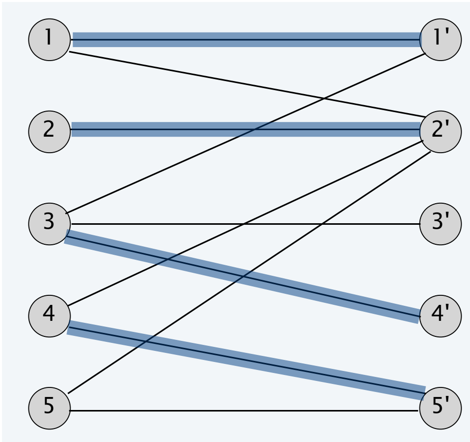
\

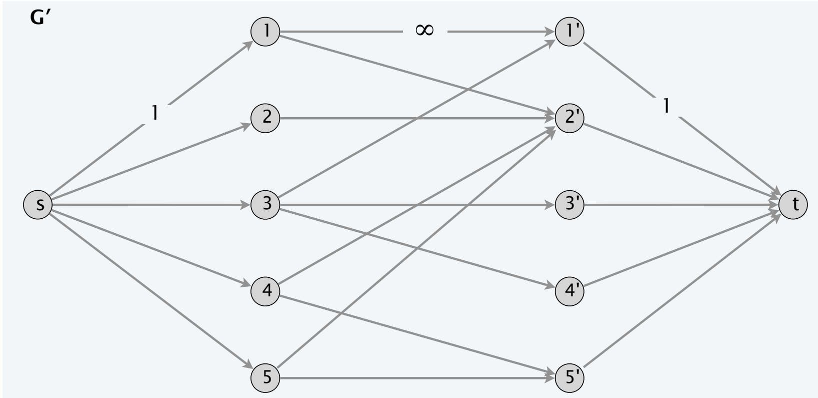
- Ballston Electronics manufactures small electrical devices.
- Products are manufactured on five different assembly lines (1,2,3,4,5).
- When manufacturing is finished, products are transported from the assembly lines to one of the five different inspection areas (A,B,C,D,E).
- Transporting products from five assembly lines to five inspection areas requires different times (in minutes)
- Cost matrix:
- What is the minimum cost assignment?

Assembly\Inspection	A	B	C	D	E
1	10	4	6	10	12
2	11	7	7	9	14
3	13	8	12	14	15
4	14	16	13	17	17
5	19	11	17	20	19

Related: Maximum Cardinality Matching

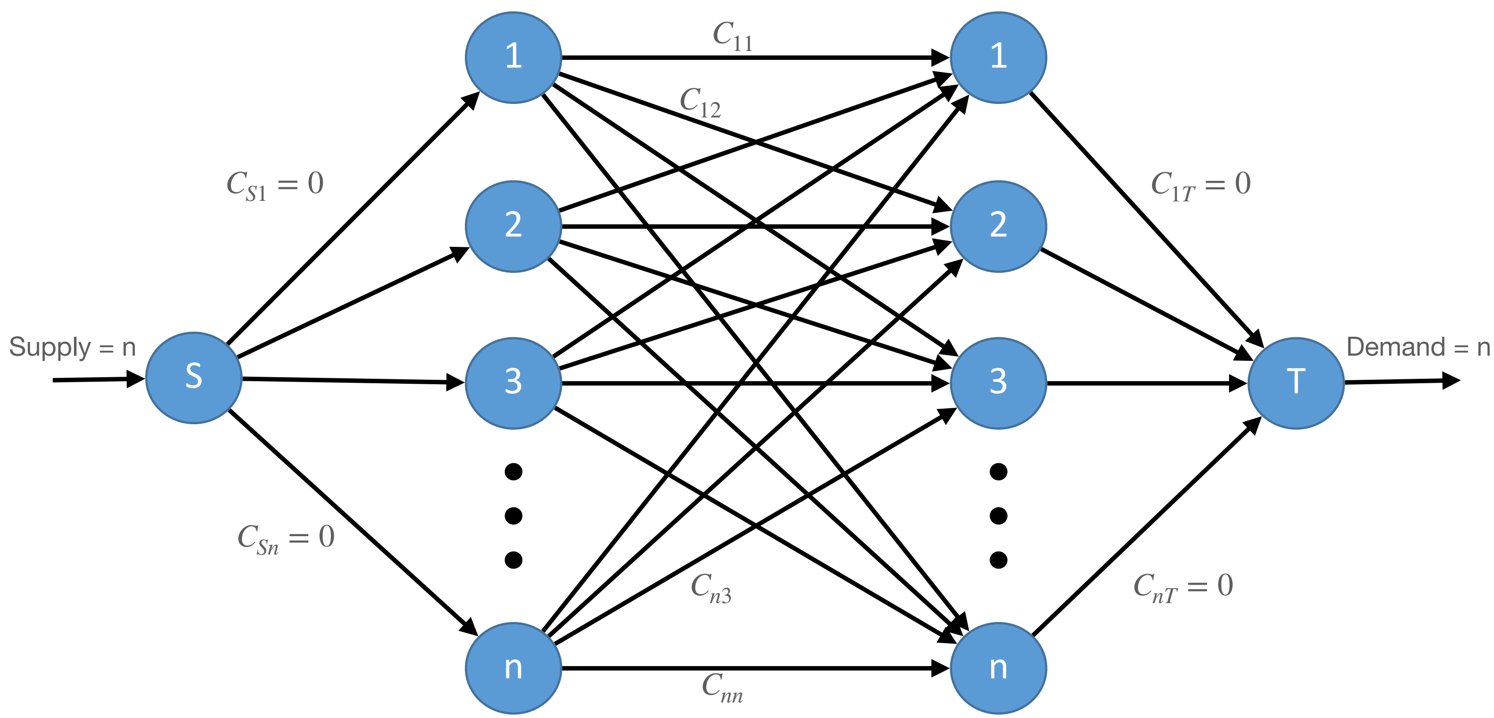
- Sparse bipartite graph $G = (V, E)$
  - ❑Edges indicate compatible matches
  - ❑Cost = 1 on all edges $e \in E$
- Solution: convert to max flow problem: assign capacity 1 to all edges, augment with vertices s, and t
- Max flow will be cardinality of maximum matching, and flow will give matches
- Solve using BFS on residual networks: shortest paths





Network Flow Representation of Assignment

- All edges capacity 1, only costs shown
- Will use for solution using combination of max-flow and shortest path



Observations

\

- Can add constant to all edges without changing optimal assignment, only changing optimal cost by a constant

$$\sum_{i=1}^{n}\sum_{j=1}^{n}(C_{ij}+K)x_{ij}=\sum_{i=1}^{n}\sum_{j=1}^{n}C_{ij}x_{ij}+\sum_{i=1}^{n}\sum_{j=1}^{n}Kx_{ij}=\sum_{i=1}^{n}\sum_{j=1}^{n}C_{ij}x_{ij}+Kn$$

- Can add object-dependent price to the cost of each edge without changing optimal assignment

$$\sum_{i=1}^{n}\sum_{j=1}^{n}(C_{ij}+p_j)x_{ij}=\sum_{i=1}^{n}\sum_{j=1}^{n}C_{ij}x_{ij}+\sum_{j=1}^{n}\sum_{i=1}^{n}p_jx_{ij}=\sum_{i=1}^{n}\sum_{j=1}^{n}C_{ij}x_{ij}+\sum_{j=1}^{n}p_j$$

- Can add person-dependent price to the cost of each edge without changing optimal assignment

$$\sum_{i=1}^{n}\sum_{j=1}^{n}(C_{ij}+q_i)x_{ij}=\sum_{i=1}^{n}\sum_{j=1}^{n}C_{ij}x_{ij}+\sum_{i=1}^{n}\sum_{j=1}^{n}q_ix_{ij}=\sum_{i=1}^{n}\sum_{j=1}^{n}C_{ij}x_{ij}+\sum_{i=1}^{n}q_j$$

Prices and Assignments

\

- Define prices $q_i, i = 1,\ldots,n$ for persons and prices $p_j, j = 1,\ldots,n$ for objects
- Define reduced costs of edges with those prices as $c^r_{p,q}(i,j') = C_{ij'} + q_i - p_{j'}$
  - ❑Note: Finding a minimum cost assignment with reduced costs results in the same assignment as finding a minimum cost assignment with original costs
- A set of prices is called admissible if $c^r_{p,q}(i,j') \geq 0$ for all $(i,j') \in E$
- A matching M is a set of assignments $\{x_{ij'}, (i,j') \in E\}$ such that $x_{ij'} \in \{0,1\}$,

$$\sum_{i:(i,j')\in E} x_{ij'} = 1 \text{ for all } j' \in 1,\ldots,n; \quad \sum_{j':(i,j')\in E} x_{ij'} = 1 \text{ for all } i \in 1,\ldots,n$$

- A matching M and a set of admissible prices $\{p,q\}$ are compatible if

$$x_{ij'} = 1 \iff c^r_{p,q}(i,j') = 0$$

Optimality Theorem

\

- If a matching M is compatible with a set of admissible prices $\{p, q\}$, then it is an optimal solution to the minimum cost assignment problem

Proof:

  - Using the admissible prices, the reduced costs of every edge is non-negative
  - The matching M has total cost 0 (all the matched edges have reduced cost 0)
  - Hence, matching M is optimal in reduced cost graph, so it is optimal in original graph

- We will exploit this concept to construct an algorithm for optimal assignments
  - Construct admissible prices and a matching M that are compatible

Successive Shortest Paths Algorithm

\

- A partial matching M' are assignments $\{x_{ij'}, (i, j') \in E\}$ such that
  $x_{ij'} \in \{0,1\}$,
  $\sum_{i:(i,j')\in E} x_{ij'} \leq 1$ for all $j' \in 1,\ldots,n$; $\sum_{j':(i,j')\in E} x_{ij'} \leq 1$ for all $i \in 1,\ldots,n$

- Idea: Given a partial matching M' compatible with a set of admissible prices $\{p, q\}$, find an augmenting path to increase the size of M' by one and modify the prices $\{p, q\}$ so the new matching is compatible with the new prices

Successive Shortest Path: Initialization

\

- Initially, set all $x_{ij'} = 0,\ (i, j') \in E$ as the initial matching $M' = \varnothing$
- Initially, set prices $p_{j'} = \min\limits_{(i,j')\in E} C_{ij'},\ q_i = 0,\ i, j' \in 1,\ldots,n$
- Note: all reduced costs $c^r_{p,q}(i, j) = C_{ij} + q_i - p_j \geq 0 \Rightarrow$ prices are admissible
- Note: initial matching M' and prices are compatible
- Construct the residual network (V, E^r) with respect to the matching M' and the prices $\{p, q\}$
  - Cost of arcs $(i, j) \in E$: $c^r_{p,q}(i, j)$
  - Cost of reverse arcs (j,i), where $(i, j) \in E$: $-c^r_{p,q}(i, j)$

Successive Shortest Path: Iteration

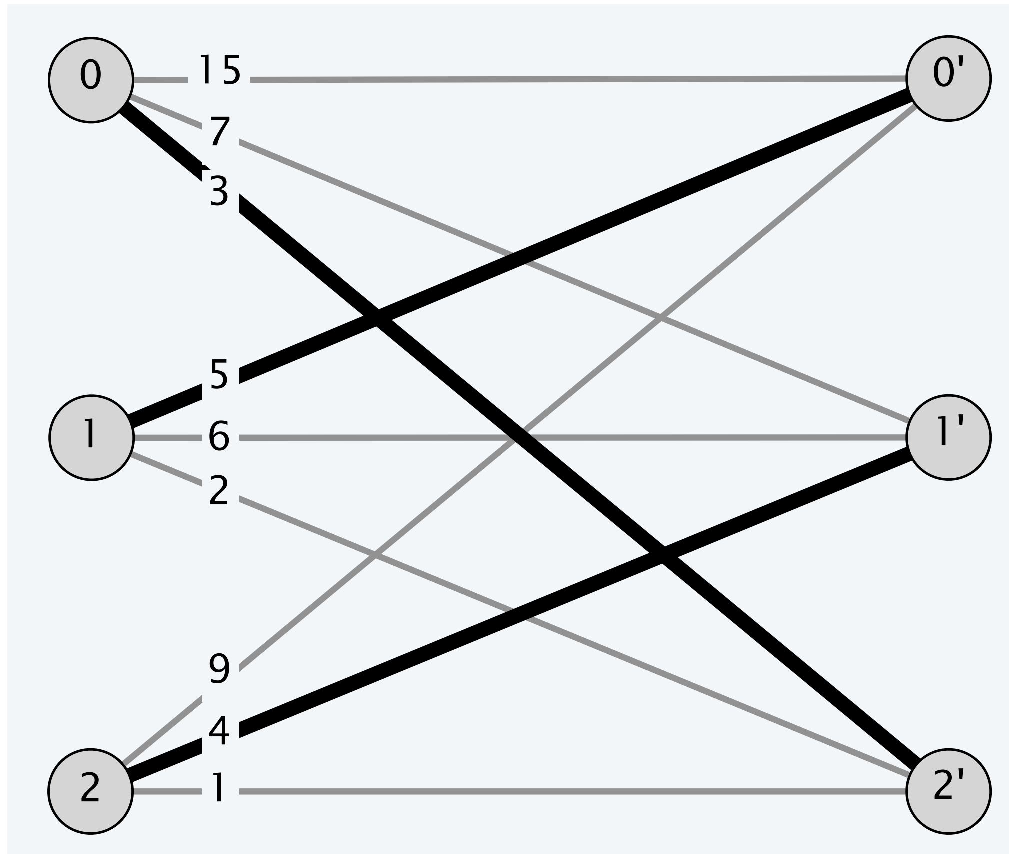
\

- Given residual network (V, E^r) with respect to the matching M' and the prices $\{p, q\}$, find shortest augmenting path P from s to t and compute the shortest distances d(i), d(j') to vertices i=1, …, n, j' = 1, …,n
- Raise prices $q_i := q_i + d(i),\ i = 1,\ldots,n; \quad p_j := p_{j'} + d(j'),\ j' = 1,\ldots,n$
- Modify assignments on augmenting path P by one unit, resulting in new partial matching M' that includes one more assignment (increases total flow to t by 1)
- Claim: new partial matching M' will be compatible with new set of admissible prices $\{p, q\}$
- Repeat above iteration n times until complete matching is found
  - Each iteration increases the number of persons matched by 1

Motivating Example

- Optimal solution:
  - ❑ $x_{02} = 1$
  - ❑ $x_{10} = 1$
  - ❑ $x_{21} = 1$

- Total cost 12

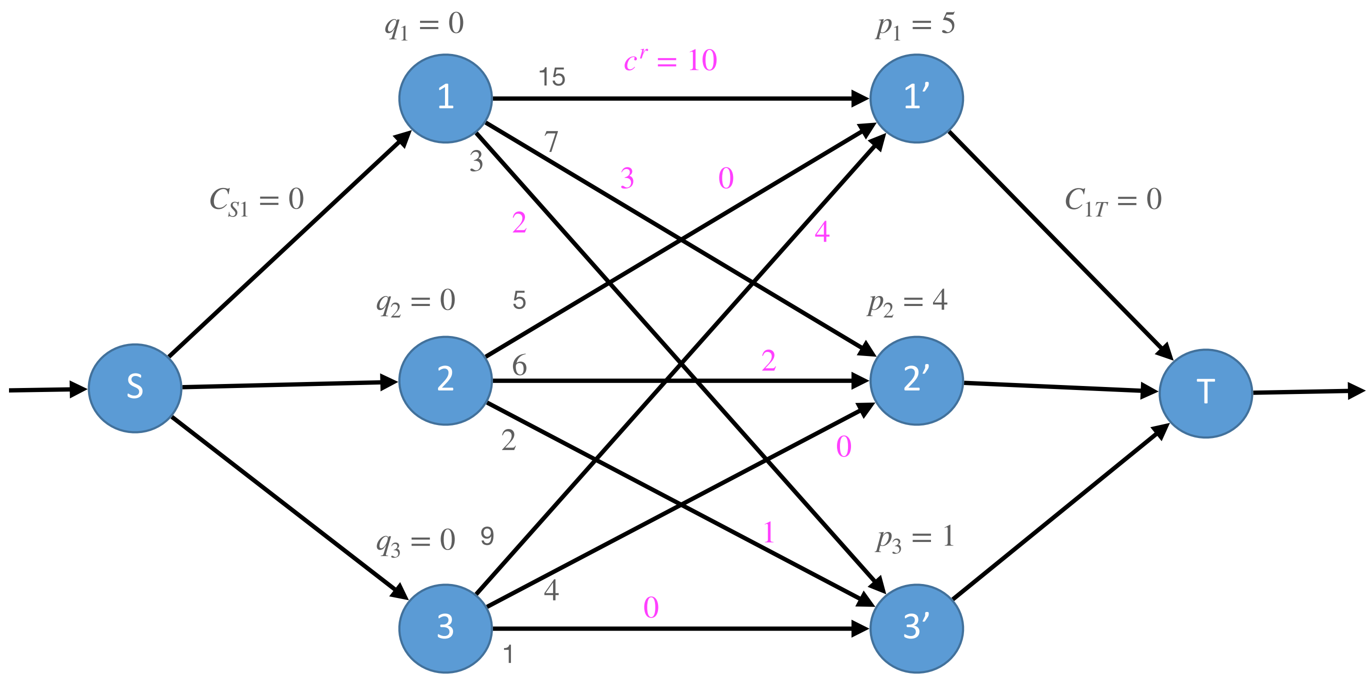


Initialization

\

Set prices $p_{j'} = \min_{i:(i,j')\in E} C_{ij'},\ q_i = 0,\ i, j' \in 1,\ldots,n$

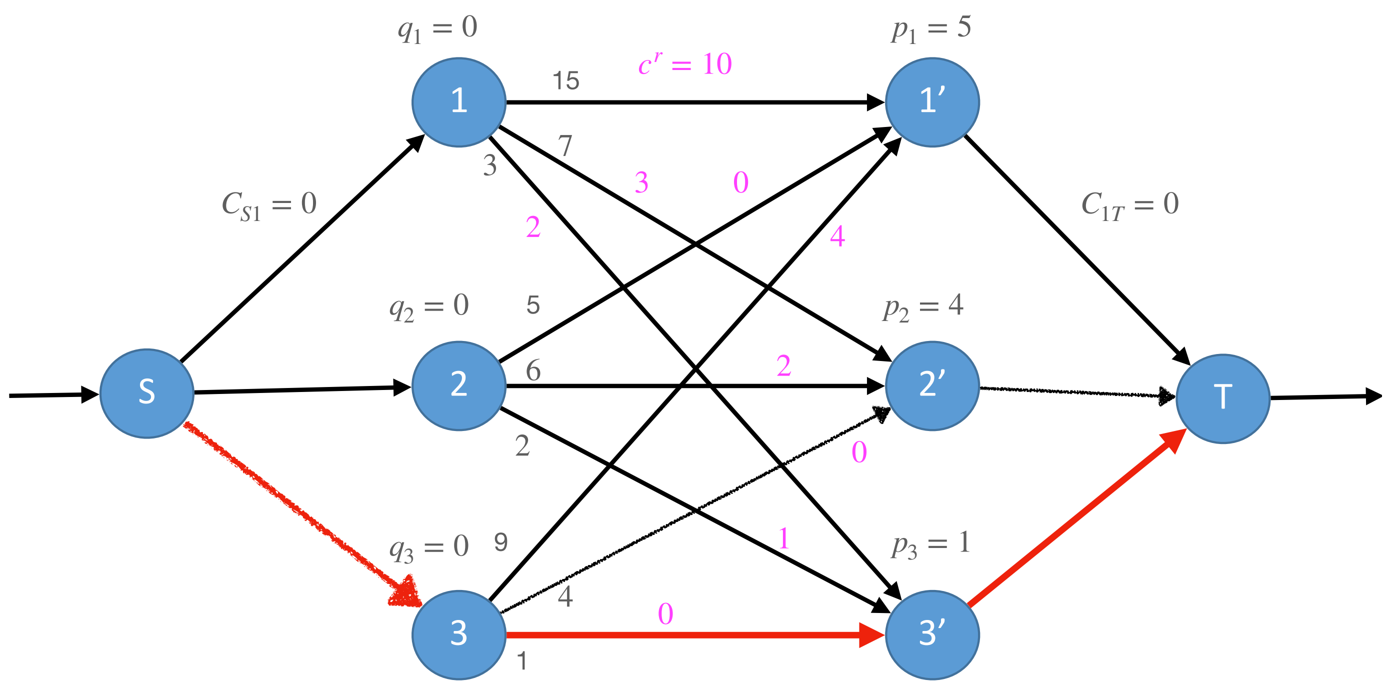
Construct residual network with reduced costs $c^r_{p,q}(i,j') = C_{ij'} + q_i - p_{j'}$



Iteration 1: Compute Shortest Path

Compute shortest path s on residual network, with distances to all vertices

Update prices q_i, $p_{j'}$ using distances



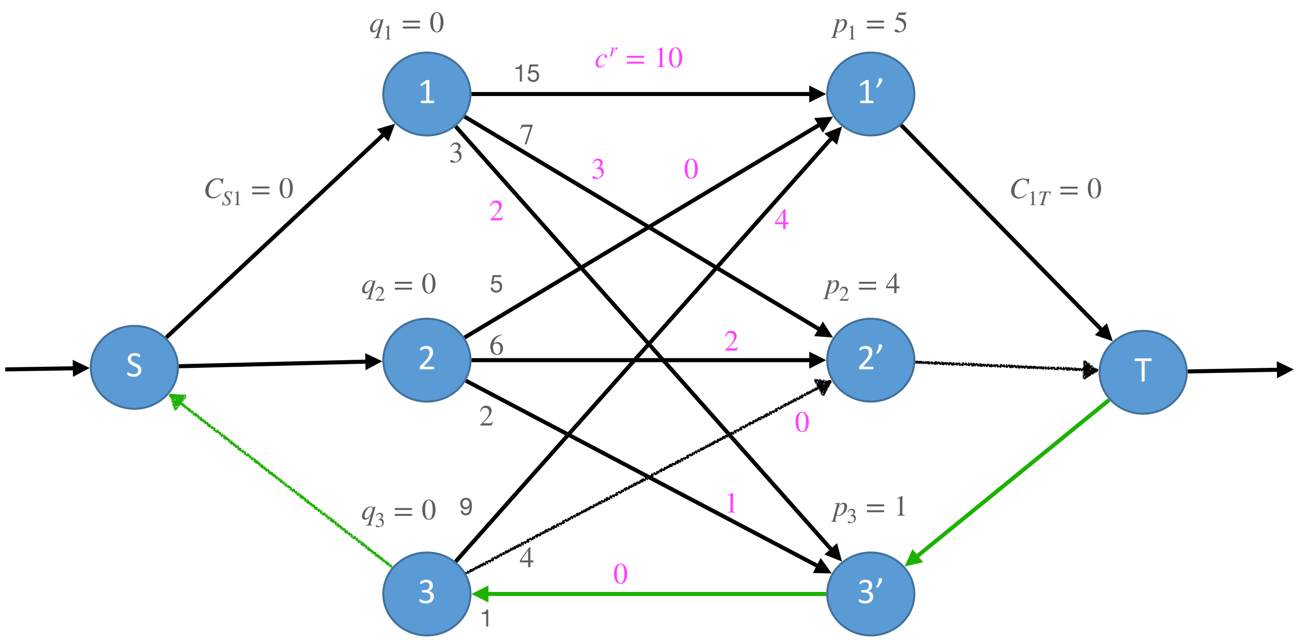
v	d(v)	p_v
1	0	0
2	0	0
3	0	0
1'	0	5
2'	0	4
3'	0	1

Perform Augmentation, Compute Residual Network

Set $x_{33'} = 1,\ x_{S3} = 1,\ x_{3'T} = 1$

Construct residual network with reduced costs $c^r_{p,q}(i,j') = C_{ij'} + q_i - p_{j'}$

Note reverse edges have non-negative reduced costs $c^r_{p,q}(j',i) = 0$

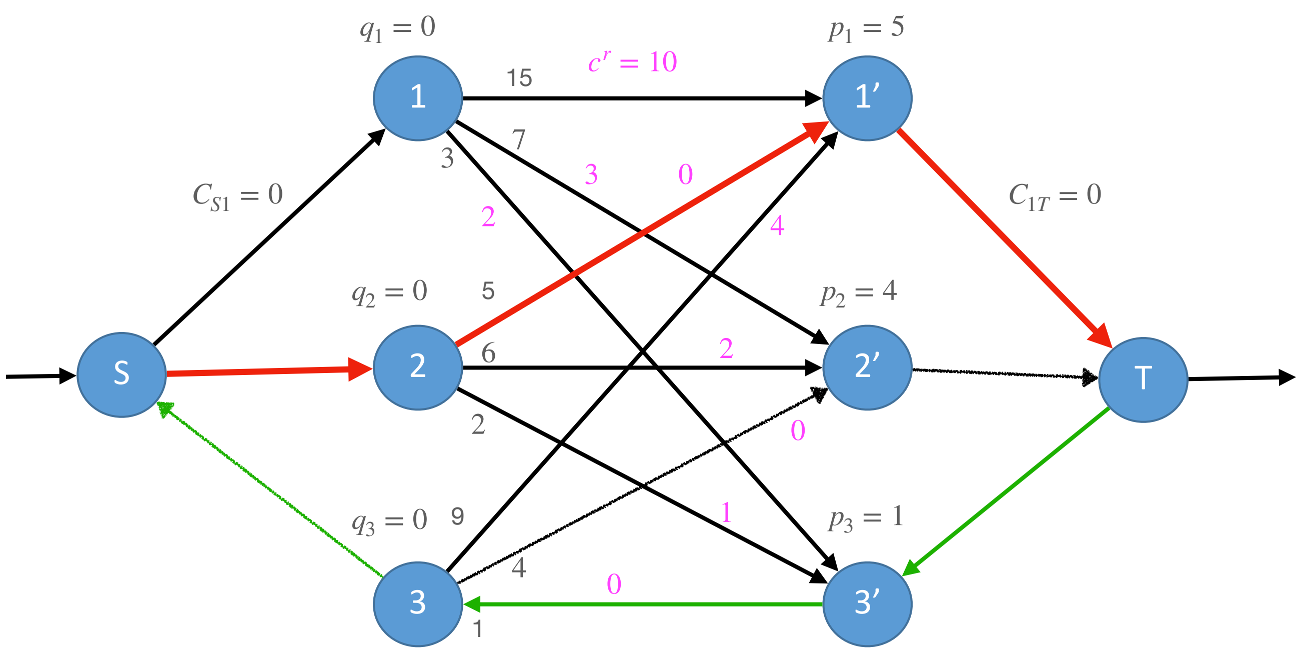


v	d(v)	p_v
1	0	0
2	0	0
3	0	0
1'	0	5
2'	0	4
3'	0	1

Iteration 2: Compute Shortest Path

Compute shortest path s on residual network, with distances to all vertices

Update prices: they stay the same, all distances are 0.



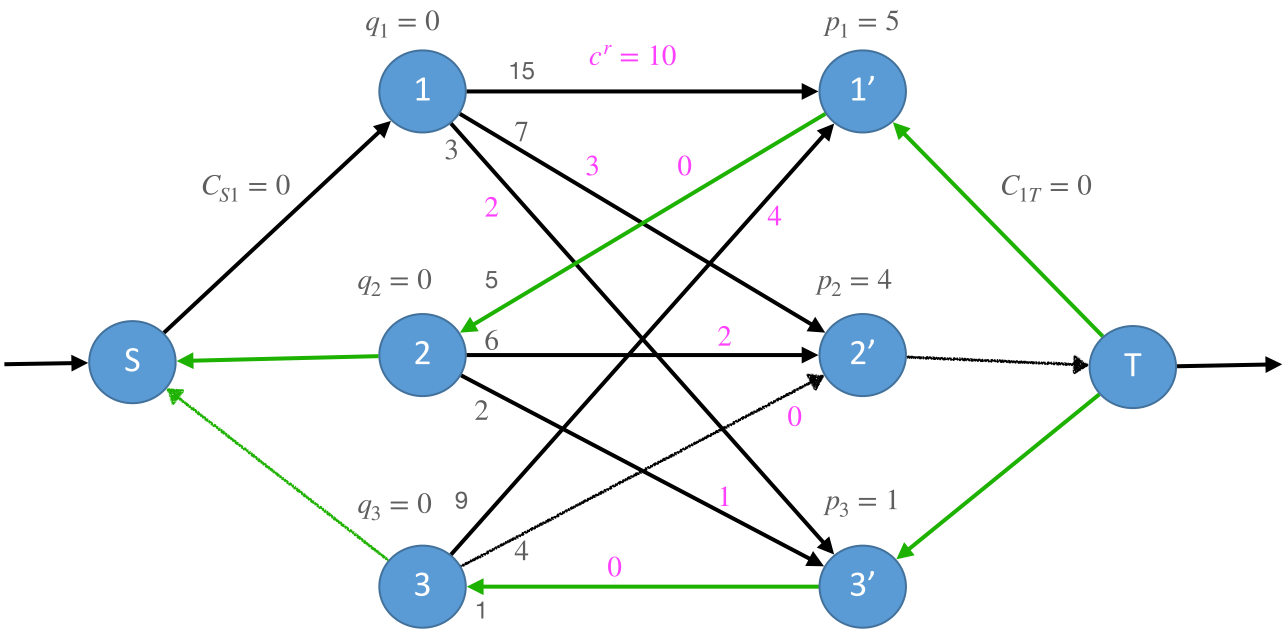
v	d(v)	
1	0	0
2	0	0
3	0	0
1’	0	5
2’	0	4
3’	0	1

Perform Augmentation, Compute Residual Network

Set $x_{21'} = 1,\ x_{S2} = 1,\ x_{1'T} = 1$

Construct residual network with reduced costs $c^r_{p,q}(i, j') = C_{ij'} + q_i - p_{j'}$

Note reverse edges have non-negative reduced costs $c^r_{p,q}(i', j) = 0$

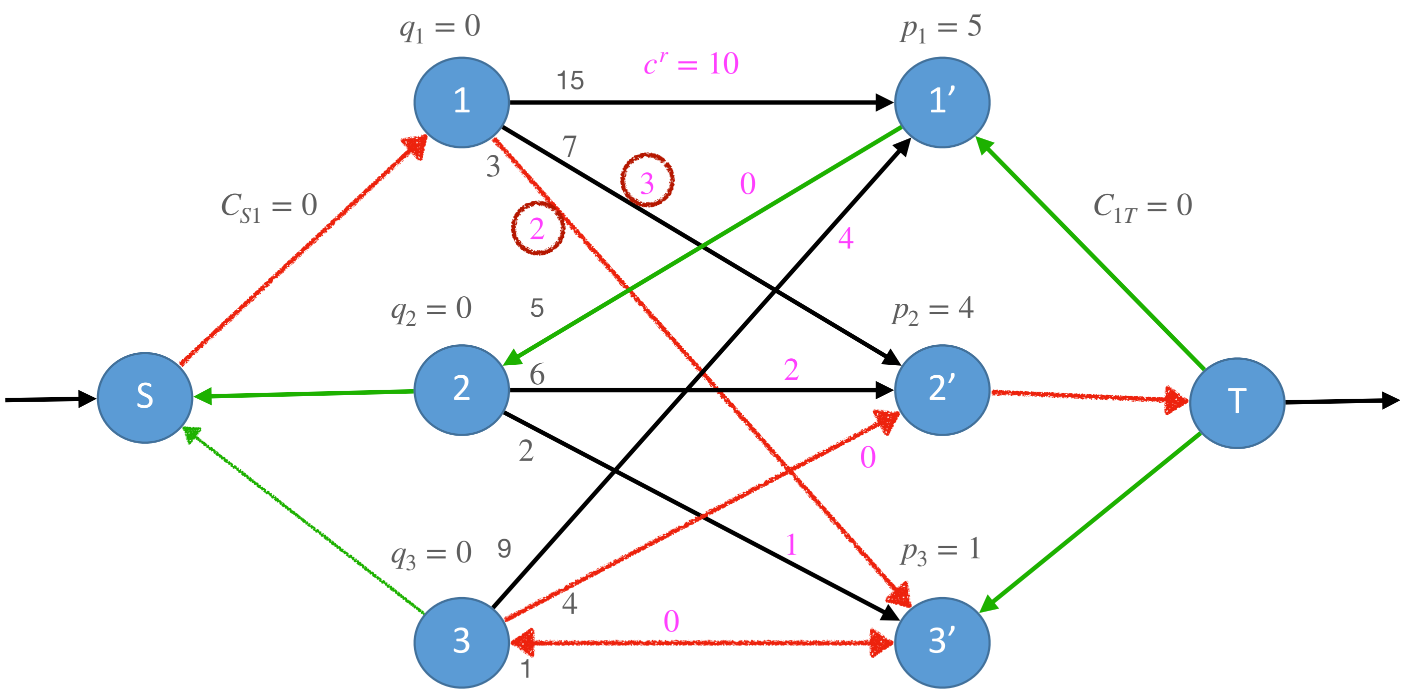


v	d(v)	p_v
1	0	0
2	0	0
3	0	0
1’	0	5
2’	0	4
3’	0	1

Iteration 3: Compute Shortest Path

Compute shortest path s on residual network, with distances to all vertices

Update prices $q_2 = 2,\ q_3 = 2,\ p_{3'} = 3,\ p_{2'} = 6,\ p_{1'} = 7$

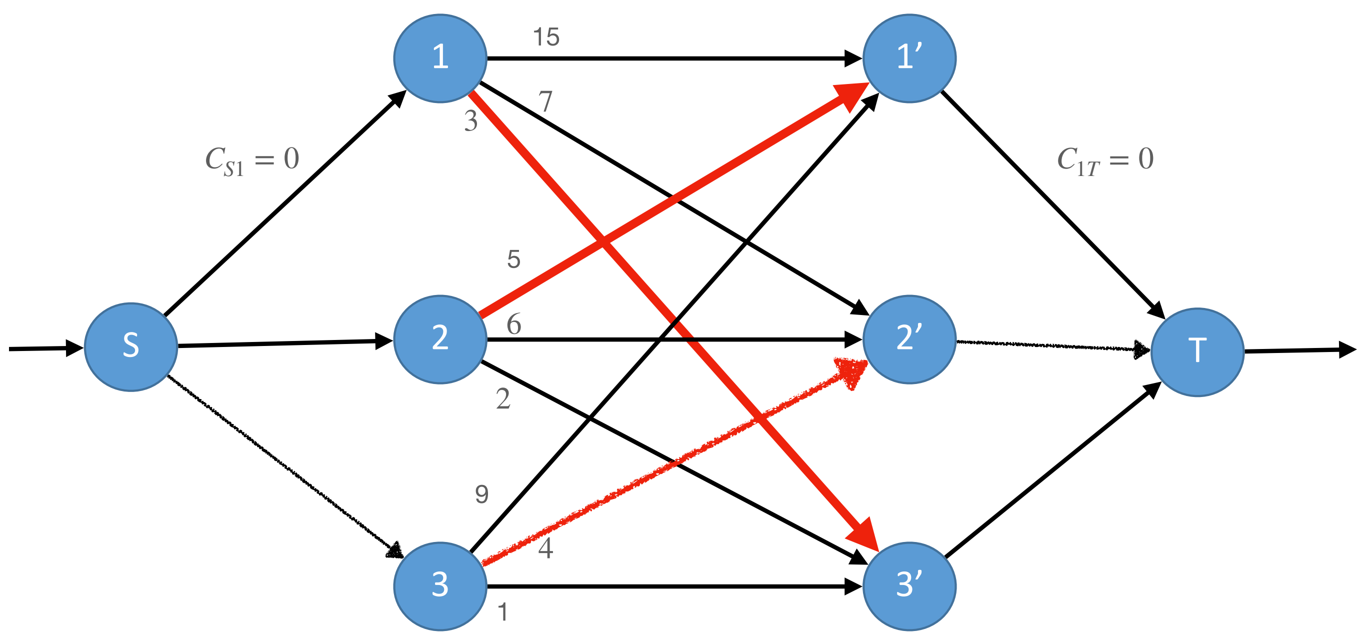


v	d(v)	p_v
1	0	0
2	2	2
3	2	2
1′	2	7
2′	2	6
3′	2	3

Perform Augmentation, Done

Set $x_{13'} = 1,\ x_{S1} = 1,\ x_{33'} = 0.\ x_{32} = 1,\ x_{2'T} = 1$

Final assignments: $x_{13'} = 1,\ x_{21'} = 1,\ x_{32'} = 1.$ Total cost = 12



v	d(v)	p_v
1	0	0
2	0	2
3	0	2
1'	0	7
2'	0	6
3'	0	3

Algorithm Complexity

\

- Can use Dijkstra's algorithm for shortest path computations
  - All reduced costs in residual networks are non-negative
  - Complexity of shortest path computation: O(|E| + |V| log(|V|)) using rank-pairing heaps or Fibonacci heaps
  - Complexity $O(|V|^2)$ using simple list priority heaps
- Complexity of augmentation and price changes: O(|V|)
- Number of iterations needed: n iterations, which is O(|V|)
- Total algorithm complexity: $O(|V|^3)$ for simple data structures

  $O\left(|V||E| + |V|^2 \log(|V|)\right)$ with best priority queues

Why Does Successive Shortest Path Work?

- Claim: at each iteration, we generate a partial matching M' and an admissible set of prices $\{p, q\}$ that are compatible
- At each iteration, the partial set of matchings contains one more matched pair than the previous iteration
- After n iterations, we obtain a complete matching M and an an admissible set of prices $\{p, q\}$ that are compatible
  - Must be optimal!

- We must show that the claim is true

Proof of Correctness

\

- Lemma: Assume we have a partial matching M' and an admissible set of prices $\{p, q\}$ that are compatible.
  - Define residual graph $G_{M'} = (V \cup \{s,t\}, E_{M'})$, reduced costs $c^r_{p,q}(i,j') = C_{ij'} + q_i - p_{j'}, (i,j') \in E$, and $c^r_{p,q}(j',i) = 0$, if $x_{ij'} = 1$ in M'
  - Define d(v) to be the shortest cost (distance) from s to vertex v in residual graph, and let P be the shortest path from s to t
  - Let $\{p', q'\}$ be the new prices after adding the distances $p'_{j'} = p_{j'} + d(j'),\ q'_i = q_i + d(i)$
- Then, every edge in P has reduced cost 0 when using prices $\{p', q'\}$
  - If $(j', i) \in P$, then $x_{ij} = 1$ in M', and $c^r_{p,q}(j',i) = 0$, d(i) = d(j'). Then,

$$c^r_{p',q'}(i,j') = C_{ij'} + q_i + d(i) - p_{j'} - d(j') = 0$$

  - If $(i, j') \in P$, then $d(j') = d(i) + C_{ij'} + q_i - p_{j'}$, and

$$c^r_{p',q'}(i,j') = C_{ij'} + q_i + d(i) - p_{j'} - d(j') = 0$$

Proof of Correctness - 2

\

- Lemma: Let $\{p, q\}$ be compatible prices for partial assignment M', and d(v) be shortest path distances in $G_{M'} = (V \cup \{s, t\}, E_{M'})$ with reduced costs $c^r_{p,q}(i, j')$, and let $\{p', q'\}$ be the new prices after adding the distances $p'_{j'} = p_{j'} + d(j'),\ q'_i = q_i + d(i)$
- Then $\{p', q'\}$ are also compatible prices for M'

Proof: if $x_{i,j'} = 1$ in M', then d(i) = d(j') and $c^r_{p,q}(j', i) = 0$, so $c^r_{p,q}(j', i) = 0$

If $x_{i,j'} = 0$ in M', then (i, j') is an edge in residual network $G_{M'} = (V \cup \{s, t\}, E_{M'})$, and $d(j') \leq d(i) + c^r_{p,q}(i, j')$ from shortest distance property, so

$0 \leq d(i) + c^r_{p,q}(i, j') - d(j) = d(i) + p(i) + C_{ij'} - d(j) - p(j) = c^r_{p',q'}(i, j')$.

Hence, $\{p', q'\}$ are admissible, and compatible with M'

Proof of Correctness - 3

\

- Lemma: Let $\{p, q\}$ be compatible prices for partial assignment M', and d(v) be shortest path distances in $G_{M'} = (V \cup \{s, t\}, E_{M'})$ with reduced costs $c^r_{p,q}(i, j')$, let $\{p', q'\}$ be the new prices after adding the distances, and let P be the shortest cost augmenting path found from s to t.
- Let M'' be the partial assignment obtained after augmenting M' with flow on path P. Then $\{p', q'\}$ are compatible prices for M''

Proof: $\{p', q'\}$ are admissible, and compatible with M'. Furthermore, for every edge (i,j') in P, $c^r_{p,q}(i, j') = 0$. Hence any $x_{ij'}$ that are set from 0 to 1 in the augmentation on P satisfy the compatibility property that $c^r_{p,q}(i, j') = 0$

- Theorem: Successive Shortest Paths results in min-cost assignment
  - We maintain the invariant that $\{p, q\}$, M' are compatible until M' is a complete assignment

Historical Notes

\

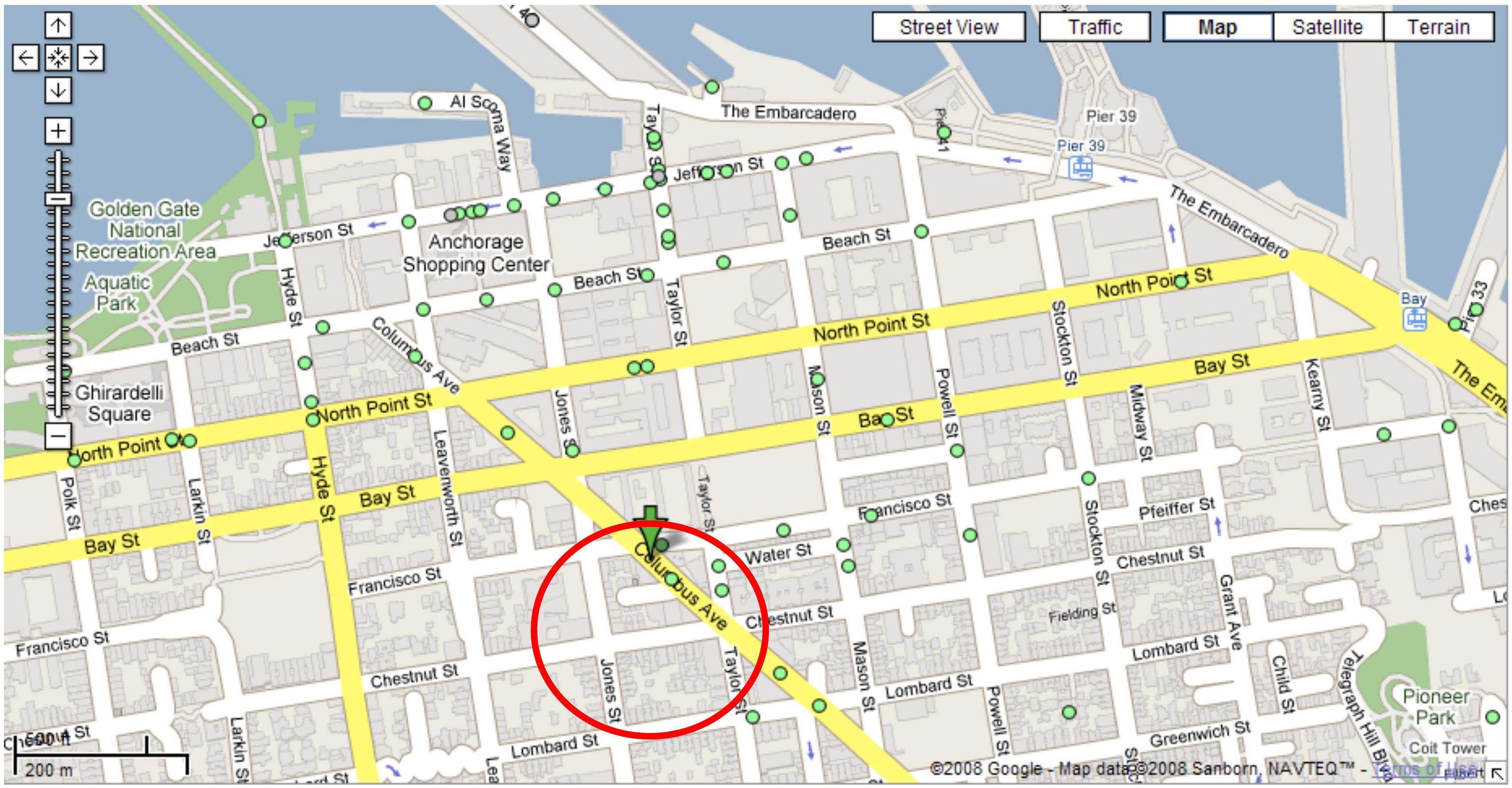
- The first algorithm for assignment is known as the Hungarian Algorithm, or Kuhn-Munkres' Algorithm, or Munkres' Algorithm
  - Hungarian name given by Kuhn in 1955 to honor ideas by König and Egerváry
  - Munkres: $O(n^4)$ algorithm (1957)
  - Edmonds-Karp: $O(n^3)$ algorithm (1971) similar to successive shortest paths
  - Fredman-Tarjan: $O\left(n^2 + n|E|\log(n)\right)$ using Fibonacci heaps
  - Bertsekas et al: (1979, 1985) Auction algorithm $O\left(n|E|\log(nC)\right)$, C is max cost
  - Orlin, others: 1991 $O\left(n^{1/2}|E|\log(nC)\right)$, combine auction and successive shortest paths
- Ideas extend to solution of more complex, non-bipartite min cost networks
  - Beyond scope of the course…

Data Structures for Multidimensional Search

- So far, focused on 1-D data
  - Balanced BSTs, B+ trees, …
- Many applications involve data which is higher-dimensional
  - Astronomy (simulation of galaxies) - 3 dimensions
  - Protein folding in molecular biology - 3 dimensions
  - Lossy data compression - 4 to 64 dimensions
  - Image processing - 2 dimensions
  - Graphics - 2 or 3 dimensions
  - Animation - 3 to 4 dimensions
  - Geographical databases - 2 or 3 dimensions
  - Web searching - 200 or more dimensions
  - Machine learning - hundreds of dimensions

K-Nearest-Neighbor

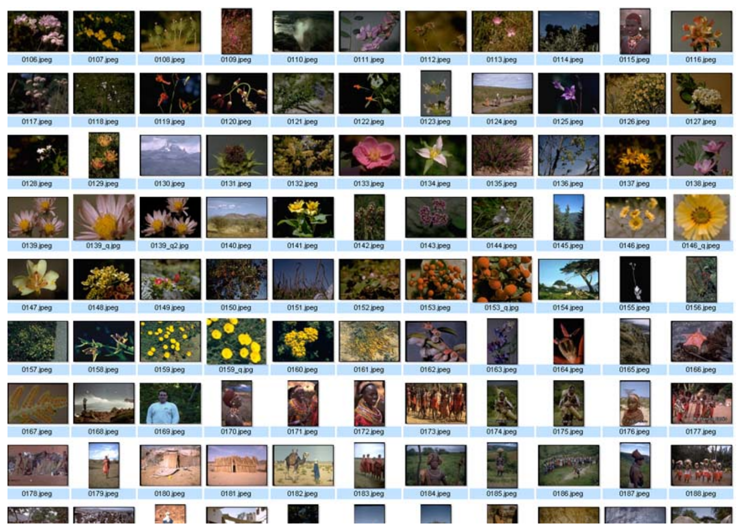
Problem: whats are the 4 closest restaurants to my hotel



Nearest Neighbor Query in High Dimensions

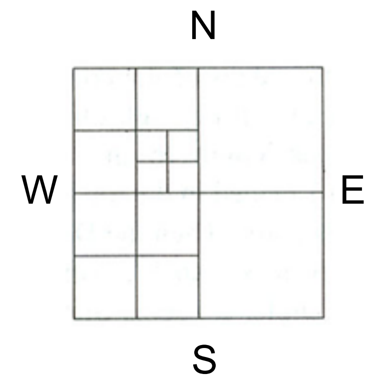
- Very important and practical problem!
  - Image retrieval

$(f_1, f_2, .., f_k)$

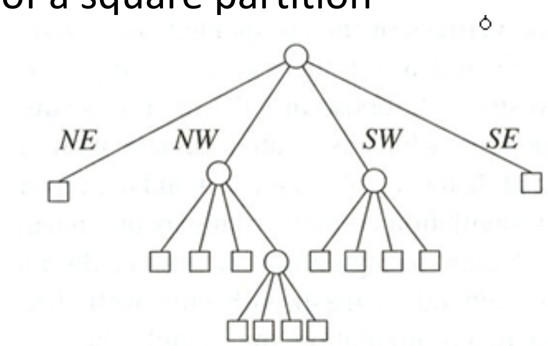


find N closest matches (i.e., N nearest neighbors)

Point-Region Quadtree



- PR Quadtrees are tries
  - ❑Trie: Decomposition based on equal division of the key space
  - ❑Shaped like a tree, with each internal node with 4 children (some empty)
- Every internal node corresponds to a region, with midpoint used for navigation
- Leaves correspond to 2-D points
- The children of a node correspond to the four quadrants of a square partition of a region
  - ❑The children of a node are labelled NE, NW, SW, and SE to indicate to which quadrant they correspond



- If a leaf contains more than one point, it splits into 4 subregions
- Need rule to break ties: arbitrary prefer N to S, E to W
- 3-D variant: Octrees