5 Greedy Algorithms

Greedy Algorithms choose next step that offers most obvious and immediate benefit, leading to locally optimal choices.

- Works for problems where making locally optimal choice leads to global optimum

Minimum Spanning Tree (MST)

Def: tree w/ min total weight on graph
Input: undirected Graph $G = (V, E)$; edge weights w_e
Output: A tree $T = (V, E')$ with $E' \subseteq E$ minimizing $weight(T) = \sum_{e \in E'} w_e$

Cut Property: On MST X with subset nodes S, for $V-S$ any lightest edge between S and $V-S$, e, is part of some MST. "Always safe to add lightest edge across any cut"

Kruskal's Algorithm:

Repeatedly add the next lightest edge that doesn't produce a cycle

```
procedure Kruskal(G,w)
  Input: connected, undirected graph G=(V,E) edge weights We
  Output: A minimum spanning tree defined by edges X
  for all u ∈ V:
     make set(u)
  X = { }
  Sort edges E by weight
  for all edges {u,v} ∈ E, in increasing order of weight:
      if find(u) ≠ find(v):
          add edge {u,v} to X
          union (u,v)
```

Runtime: $O(|E| \log |V|)$

Prim's Algorithm:

Intermediate set forms subtree and grows by one each iteration

```
procedure prim(G, w):
    for all u ∈ V:
        cost(u) = ∞
        prev(u) = nil
    Pick any initial node u_0
    cost(u_0) = 0

    H = makequeue(v)
    while H is not empty:
        v = deletemin(H)
        for each {v,z} ∈ E
            if cost(z) > w(v,z):
                cost(z) = w(v,z)
                prev(z) = v
                decreasekey(H,z)
```

Runtime: $O(|E| \log|V|)$

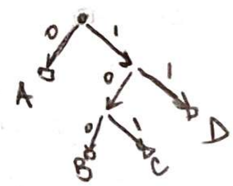
5.2

Huffman Encoding

Want to encode data efficiently

- Use variable length encoding, prefix free, binary tree to represent
- Symbols w/ smallest freq at bottom of tree,

letter	encoding
A	0
B	100
C	101
D	11




```
procedure huffman(f):
    Input: An array f[1,...n] of frequencies
    Output: An encoding tree w/ n leaves

    let H be a priority queue of integers, ordered by f
    for i=1 to n: insert(H,i)
    for k=n+1 to 2n-1
        i = deletemin(H), j = deletemin(H)
        create node numbered k w/ children i,j
        f[k] = f[i] + f[j]
        insert(H, k)
```

Runtime: $O(n \log n)$

Set Cover

Given set of points, want to find smallest num of subsets to cover all points

Set Cover Algorithm:

Input: A set of elements B, sets $S_1, \dots, S_m \subseteq B$

Output: A selection of the S_i whose union is B

Cost: number of sets picked

Repeat until all elements of B are covered

Pick the set S_i w/ largest number of uncovered elements

If optimal k sets, greedy algo returns at most $k \ln n$ sets

6) Dynamic Programming

Solve problem by finding subproblems, tackling them smallest first and using answers from smaller problems to solve larger ones

Shortest Paths in DAGs

Compute in single pass

initialize all dist(·) values to 0

$dist(s) = 0$

for each $v \in V \setminus \{s\}$, in linearized order:

$dist(v) = \min_{(u,v)\in E} \{dist(u) + l(u,v)\}$

Longest Increasing Subsequence : $O(n^2)$

find longest increasing subsequence in list

for $j = 1, 2, \dots, n$:

$L(j) = 1 + \max\{L(i) : (i,j) \in E\}$

return $\max_j L(j)$

Min Edit Distance : $O(mn)$

Edit distance between two words, remove, add, change

for $i = 0, 1, 2, \dots m$:

$E(i,0) = i$

for $j = 1, 2, \dots n$:

$E(0,j) = j$

for $i = 1, 2, \dots m$:

for $j = 1, 2, \dots n$:

$E(i,j) = \min\{E(i-1,j)+1,\ E(i,j-1)+1,\ E(i-1,j-1) + diff(i,j)\}$

return $E(m,n)$

Runtime: $O(mn)$

Knapsack : $O(nW)$

Bag fits at most W weight each object weight $w_1, w_2 \dots w_n$ dollar value $v_1, \dots v_n$

① Unlimited quantity : subproblem by knapsack capacity w

$K(0) = 0$

for $w = 1$ to W:

$K(w) = \max\{K(w - w_i) + v_i : w_i \le w\}$

return $K(W)$

② No repetition: capacity w and keep track of items left

Initialize all $K(0,j) = 0$ and $K(w,0) = 0$

for $j = 1$ to n:

for $w = 1$ to W:

if $w_j > w$: $K(w,j) = K(w, j-1)$

else: $K(w,j) = \max\{K(w, j-1),\ K(w - w_j, j-1) + v_j\}$

return $K(W, n)$

All pairs shortest paths

Find shortest paths between all s,t

Floyd-Warshall Algorithm : $O(|V|^3)$

Start from one node expand set of intermediate nodes

```
for i=1 to n:
    for j=1 to n:
        dist(i,j,0) = ∞

for all (i,j) ∈ E:
    dist(i,j,0) = l(i,j)

for k=1 to n:
    for i=1 to n:
        for j=1 to n:
            dist(i,j,k) = min { dist(i,k,k-1) + dist(k,j,k-1),
                                dist(i,j,k-1) }
```

Independent sets in trees : $O(|V|+|E|)$

Find independent set where no edges between nodes

$I(u)$ = size of largest independent set from u

Either includes or doesn't

$$I(u) = \max\left\{1 + \sum_{\substack{\text{grandchildren}\\ w \text{ of } u}} I(w),\ \sum_{\substack{\text{children } w\\ \text{of } u}} I(w)\right\}$$

if independent set, can't include children.

7 Linear Programming and reductions

Optimization tasks where constraints and optimization criterion are linear functions

Optimum typically achieved at a vertex of feasible region, or infeasible (too tight constraints), unbounded (too loose)

- Can be solved with Simplex method to start at vertex repeatedly looking for better obj value

Primal

$\max \quad c_1x_1 + \dots + c_nx_n$

$a_{i1}x_1 + \dots + a_{in}x_n \le b_i \quad i \in I$

$a_{i1}x_1 + \dots + a_{in}x_n = b_i \quad i \in E$

$x_j \ge 0 \quad j \in N$

$\max\ c^Tx$
$Ax \le b$
$x \ge 0$

Dual

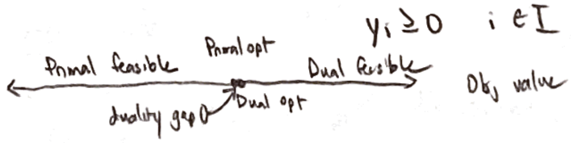
$\min \quad b_1y_1 + \dots + b_my_m$

$a_{1j}y_1 + \dots + a_{mj}y_m \ge c_j \quad j \in N$

$a_{1j}y_1 + \dots + a_{mj}y_m = c_j \quad j \notin N$

$y_i \ge 0 \quad i \in I$

$\min\ y^Tb$
$y^TA \ge c^T$
$y \ge 0$



Constraint Transformations:

① Changing Objective:

$\max\ c^Tx = \min\ -c^Tx$

$\min\ c^Tx = \max\ -c^Tx$

② Inequality to Equality

$ax \le b \rightarrow ax + s = b,\ s \ge 0$

③ Equality to Inequality

$ax = b \rightarrow ax \le b,\ ax \ge b$

④ Unrestricted variable

$x \in \mathbb{R} \rightarrow x = x_+ - x_-$

$x_+, x_- \ge 0$

Ex) Chocolates to max profit

Objective: $\max\ x_1 + 6x_2$

Constraint: $x_1 \le 200$
$x_2 \le 300$
$x_1 + x_2 \le 400$
$x_1, x_2 \ge 0$

LP: use Simplex

$(x_1, x_2) = (100, 300)$

Dual:

Multiplier	Inequality
y_1	$x_1 \le 200$
y_2	$x_2 \le 300$
y_3	$x_1 + x_2 \le 400$

$(y_1+y_3)x_1 + (y_2+y_3)x_2 \le 200y_1 + 300y_2 + 400y_3$

$\min\ 200y_1 + 300y_2 + 400y_3$

$y_1 + y_3 \ge 1$

$y_2 + y_3 \ge 6 \qquad y_1, y_2, y_3 \ge 0$

$(y_1, y_2, y_3) = (0, 5, 1)$

$5x_2 \le (300)5$

$x_1 + x_2 \le 400$

$x_1 + 6x_2 \le 1900$

$(x_1, x_2) = (100, 300)$

$100 + 6(300) \le 1900$

Zero Sum Games

Can represent some situations with matrix games with payoff matrix

- row wants to maximize and col to minimize
- Each player can have a mixed strategy to decide each play with probability x_i
- Expected (avg) payoff is $\sum_{i,j} G_{ij} \cdot \text{Prob}[\text{row plays } i, \text{col plays } j]$

		Column		
		r	p	s
row	r	0	-1	1
	p	1	0	-1
	s	-1	1	0

In cases, where opponent move is known, want to play defensively

For row:

(x_1, x_2) to max $\underbrace{\min\{3x_1 - 2x_2, -x_1 + x_2\}}_{\text{columns best response}}$

	m	t
e	3	-1
s	-2	1

$z = \min\{3x_1 - 2x_2, -x_1 + x_2\}$

max z
$z \le 3x_1 - 2x_2$
$z \le -x_1 + x_2$

max z

$-3x_1 + 2x_2 + z \le 0$
$x_1 - x_2 + z \le 0$
$x_1 + x_2 = 1$
$x_1, x_2 \ge 0$

Both have same optimum

For column. min max $\{3y_1 - y_2, -2y_1 + y_2\}$

min w

$-3y_1 + y_2 + w \ge 0$
$2y_1 - y_2 + w \ge 0$
$y_1 + y_2 = 1$
$y_1, y_2 \ge 0$

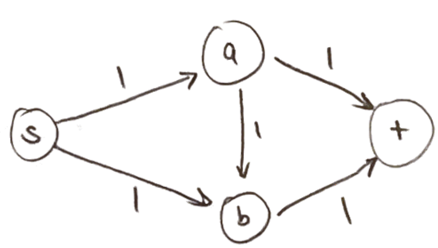
$$\max_x \min_y \sum_{i,j} G_{ij} x_i y_j = \min_y \max_x \sum_{i,j} G_{ij} x_i y_j$$

Max Flow

For graph, want to send as much flow through, each edge has capacity

- from source s to target t
- The shipping scheme is flow w/ f_e on every edge, must:

1. Doesn't violate edge capacities: $0 \le f_e \le c_e$ for all $e \in E$
2. All nodes u except s, t flow entering equals flow leaving



flow conserved $\sum_{(w,u) \in E} f_{wu} = \sum_{(u,z) \in E} f_{uz}$

Size of flow = quantity leaving s $\sum_{(s,u) \in E} f_{s,u}$

Ford Fulkerson Algorithm

Start with zero flow

Find path from s to t and route max flow through

Update capacities by updating residual graph

Repeat until no more paths s to t

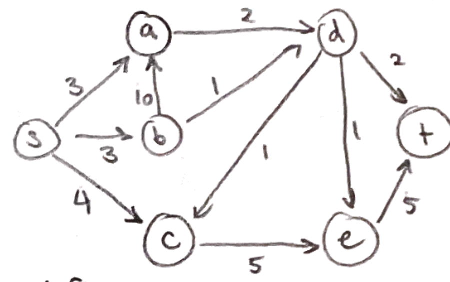
Residual Graph

Two types of edges   residual capacity c^f.

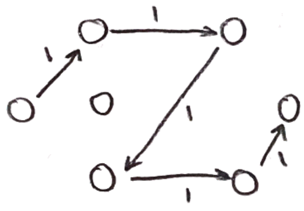
$c_{uv} - f_{uv}$ if $(u,v) \in E$ and $f_{uv} < c_{uv}$

f_{vu} if $(v,u) \in E$ and $f_{vu} > 0$

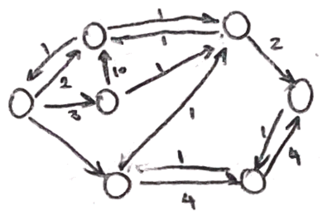
Graph



flow



residual



forward: capacity left

back: flow through edge

For any (s,t) cut on graph, $size(f) \leq capacity(L,R)$

capacity of cut

Max-flow min-cut theorem

Size of max flow in a network equals capacity of smallest (s,t) cut

Max flow creates residual graph, using graph, find min cut by finding vertices(L) reachable from s and V−L separation is the min cut in graph

Runtime: $O(|V| \cdot |E|^2)$