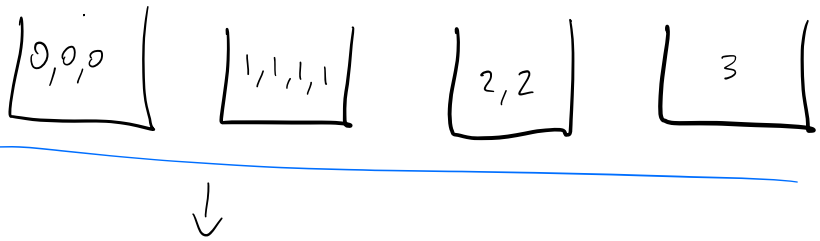
Lecture 12: Sorting Wrap Up / Graphs

Bucket Sort: If only k "types" appear in your list, make k "buckets"

[1, 0, 1, 0, 2, 2, 1, 3, 1, 0]



Step 1: place into buckets

$O(n)$

Step 2:

$O(n+k)$

[0, 0, 0, 1, 1, 1, 1, 2, 2, 3]

In place? NO

Adaptive? NO

Stable? YES

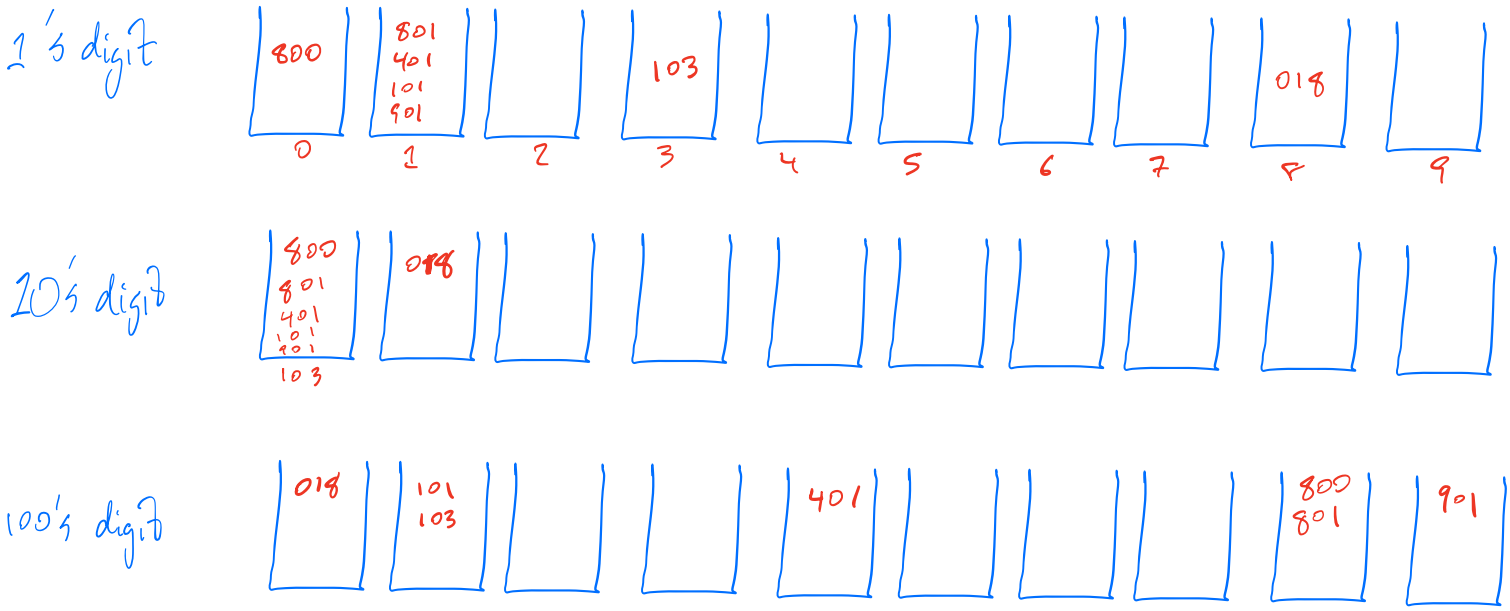
Radix Sort

radix = base of number

idea: repeatedly apply bucket sort to the digits of elements in your list, from least to most significant.

Example: (base/radix = 10)

[103, 801, 802, 401, 101, 901, 018]



→ [018, 101, 103, 401, 800, 801, 901].

Key property of BucketSort needed? BucketSort is stable

runtime? depends on radix/base b and n and K ($\leq K(n)$)

n: # elements

K: max size of integer

$n \log_b K$

# of buckets = b

Bucket sort $\log_b K$ times

$n+b$

$O((n+b)\log_b K)$

example: suppose we know each element of our size n list is a positive integer smaller than $n^{100\sqrt{n}}$.

K = # of buckets

Bucket?

$O(n+K)$

$O(n + n^{100\sqrt{n}}) = O(n^{100\sqrt{n}})$ ≠ ~~$O(n\sqrt{n})$~~

comparison-based? $O(n \log n)$

$n^{100\sqrt{n}} = 2^{100\sqrt{n}\log n}$

radix? base 2: # of buckets $= 2$

# of bucketrounds? $\log_2 n^{100\sqrt{n}} = 100\sqrt{n}\log n$

base n: $O(n \cdot \sqrt{n})$

$O\left(n^{3/2}\log n\right)$

ex 2| $\leq n^{100\sqrt{\log n}}$

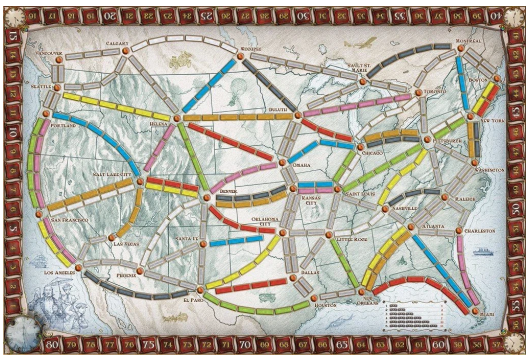
bucketsort: $O\left(n + n^{100\sqrt{\log n}}\right) = O\left(n^{100\sqrt{\log n}}\right)$

$2^{100\log^{3/2} n}$

radix? base 2: $O((n+2)\cdot \log^{3/2} n)$

base n: $O(n+n)\,100\sqrt{\log n}$

GRAPHS



Def (Undirected) $G = (V, E)$

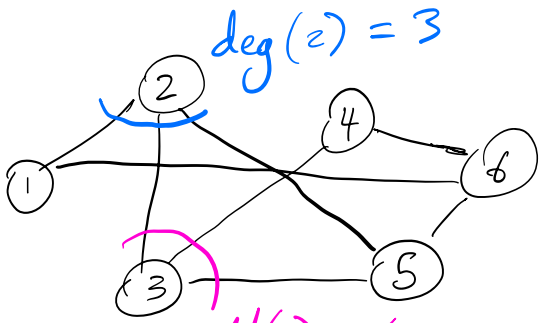
V = set of vertices/nodes

$V = \{1,2,3,4,5,6\}$

$E = \{(1,2), (1,6), (4,6), \ldots\}$

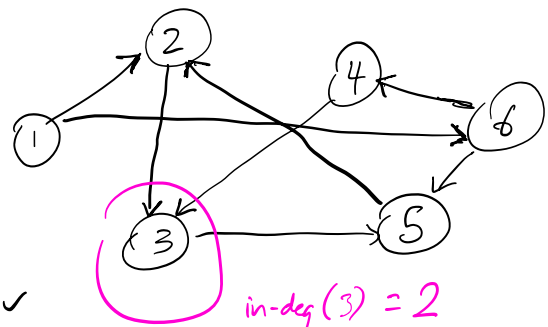
degree $\deg(v)$ = # of edges touching v.

neighbors/adjacents: $N(v)$ = all nodes connected with single edge to v



Directed: $G = (V, E)$

edges are ordered

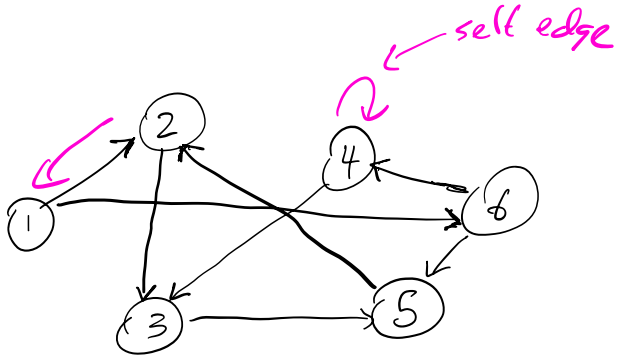


"in-degree" = # of edges coming "in" to v

"out-degree" = " " "out" of v

non-simple graphs

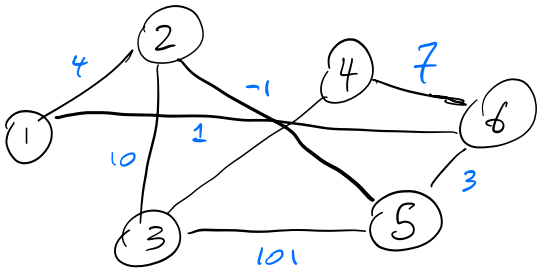
a graph is "simple" if it has no duplicate or self edges



weighted graphs

weight function

$w: E \to \mathbb{R}$



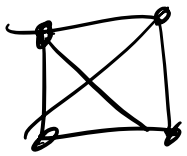
Applications?

what are nodes/edges? directed? simple? weighted?

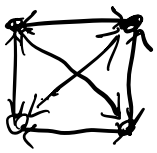
- facebook friends — nodes: accounts — edges: friends? — undirected
- Java Inheritance — nodes = classes — edges = inheritance? — weighted?
- Airline/train routes
- Bus stations/stops
- Neural nets.
- google maps.
- Wikipedia.

Complete Graphs

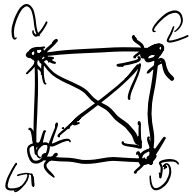
undirected



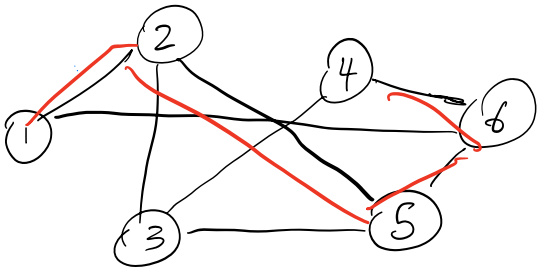
directed



directed, non-simple



Def (path) a sequence of vertices s.t. there is an edge for every pair of vertices in the sequence



$$p = (1, 2, 5, 6, 4)$$

simple path no vertex is repeated in path.

cycle path starts and ends on the same vertex.

$$p = (2, 5, 6, 4, 3, 2)$$

"six degrees of separation" in facebook graph, $\forall\, v_1, v_2$ $\exists$ path from v_1 to v_2 of length at most 6

Def (strongly connected) directed $G = (V, E)$ is strongly connected if $\forall\, v_1, v_2 \in V$ $\exists$ path p starting at v_1 and ending at v_2.

not strongly connected

is weakly connected

weakly connected for a directed G, G is "weakly connected" if when you take away directionality of edges, $\tilde{G}$ is connected.

maximum # of edges?

undirected, simple :

directed, simple :

dense:

sparse:

question: if G connected, min # of edges?

In general, $|E|$ and $|V|$ very different, so runtime should be a function of both $|E|$ and $|V|$

Trees