INPUT A | 3 1 4 3 5 4 3 5 1 1 4 4 |

B[3, 0, 3, 4, 2] ↓

OUTPUT | 1 1 1 3 3 3 4 4 4 4 5 5 |

COUNTING SORT

INPUT: $A[1 \ldots n]$ $A[i] \in [m]$

time $O(n+m)$ ✓

$B[1 \ldots m]$

for $i = 1$ to m
  $B[i] := 0$
for $j = 1$ to n
  $B[A[j]]{+}{+}$ increment $A[j]$ entry of B

(: after this loop $B[k]$ counts occurrences of k in A :)

then expand B to output (DO: pseudocode)

RADIX SORT

CAA	CAA	CAA	AAB
BCB	ACA	CAA	ACA
BBB	CAA	AAB	BBB
ACA	BCB	BBB	BCB
AAB	BBB	ACA	BCB
BCB	AAB	BCB	CAA
CAA	BCB	BCB	CAA

words: length k

words: ℓ

letters: m

---

each round costs $O(\ell + m)$

counting sort

total $O(k(\ell+m))$

for fixed alphabet: m = const

$O(k\ell)$ "linear time"

$k\ell$ size of input

REWARD

CAA . .
BBCB
ABABABABAABCA

Sort list of words of variable length in linear time if alphabet fixed

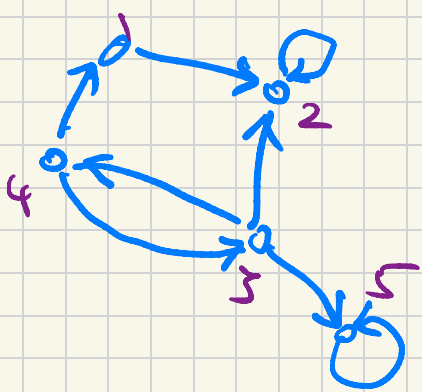
---

Digraphs (directed graphs)

$G = (V, E)$

$E \subseteq \binom{V}{2}$

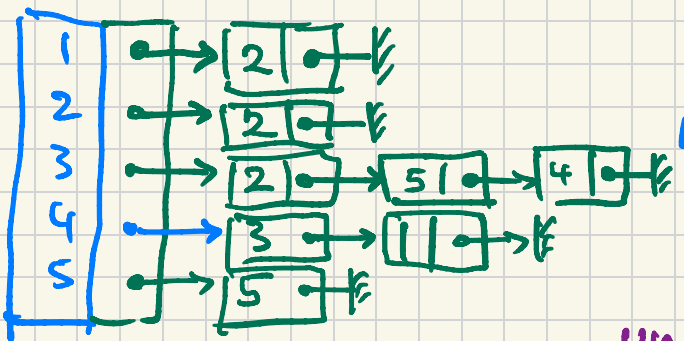
$E \subseteq V \times V$



$E = (12, 22, 32, 35, 55, 34, 43, 41)$

edge list

adjacency array:
array of adjacency lists



array

Unit cost model

input size

$n + m$

$n = |V|$ $m = |E|$

for $i \in V$

$A[i]$: linked list out neighbors

"single pass" scheme

for $i \in V$

  for $j \in adj(i)$

    do something

"LINEAR TIME"

$O(n+m)$ steps

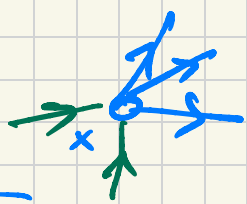
(undirected) graph represented as a digraph:

$\{i, j\}$ replace $(i, j), (j, i)$

---

out degree $\deg^+(x)$

in degree $\deg^-(x)$



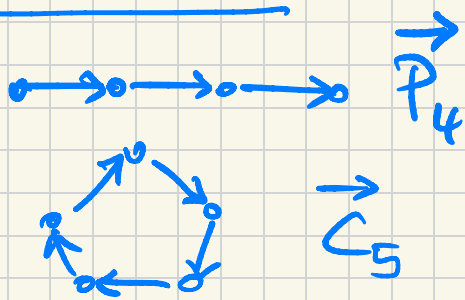
$$\sum_{x \in V} \deg^+(x) = \sum_{x \in V} \deg^-(x) = m$$

m = # edges

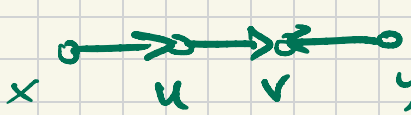
---

directed path $\vec{P}_4$

" cycle $\vec{C}_5$



y is accessible from x if $\exists\ x \to \cdots \to y$ directed path



accessibility

- reflexive
- transitive
- NOT symmetric

DEF digraph G is strongly connected

if $(\forall x, y \in V)$(y is accessible from x)

$\overrightarrow{dist}(x, y)$: length of shortest $x \to \cdots \to y$ directed path

DEF walk (allows repeated vertices)

DO: If $\exists x \to \cdots \to y$ walk then $\exists x \to \cdots \to y$ path

---

GOAL given "root" r

for all $x \in V$

find $\overrightarrow{dist}(r, x)$

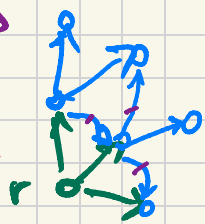
Breadth-first search BFS

r : root

reach all vertices from root

create parent links → BFS tree

rooted tree



Q: FIFO queue
discovered, waiting to be explored

initialization

for $x \in V$
  status(x) = WHITE
  dist(x) = ∞
  p(x) = NIL ← parent link: no parent

Q : empty FIFO queue (: always: currently explored vertices:)

enqueue(Q, r) (: root discovered :)
  status(r) := GREY
  dist(r) := 0
  p(r) := r ← parent link

main loop

while Q not empty
  x ← dequeue(Q) (: begin exploring x)
  for $y \in adj[x]$
    if status(y) = WHITE then
      status(y) := GREY (: y discovered)
      enqueue(Q, y)
      p(y) := x ← parent link: y discovered from x
      dist(y) := dist(x) + 1
  status(x) := BLACK (: x completed)