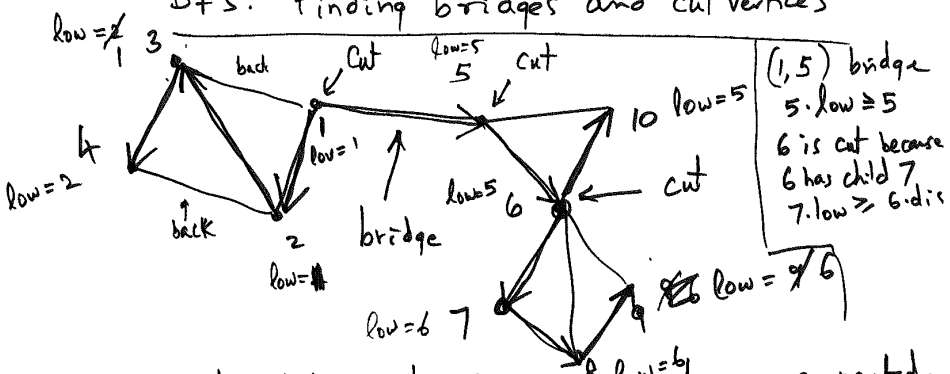
DFS: Finding bridges and cut vertices



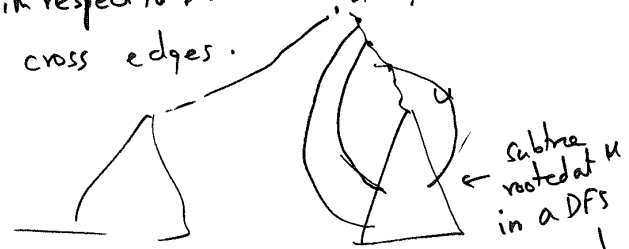
Input: Undirected graph $G=(V,E)$ — Assume connected.

Edge $e=(u,v)$ is a bridge if removing (u,v) from G disconnects G.

Vertex u is a cut vertex if removing u (and its edges) from G disconnects it.

Facts on DFS

1. Edges of G with respect to DFS — tree / back edges.
   There are no cross edges.



2. Only tree edges can be bridges.
   Is $(u, u.parent)$ a bridge?

Compute $u.low$ for each vertex $u \in V$.

Rule for finding bridges:

$(u, u.parent)$ is a bridge if $u.low \geq u.dis$

Rule for finding cut vertices:

(i) u = root of DFS is a cut vertex if
u has 2 or more children in DFS tree

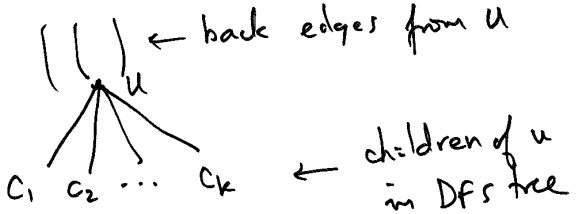
(ii) $u \neq$ root of DFS tree is a cut vertex if
u has a child c such that $c.low \geq u.dis$

5 is a cut vertex because
5 has child 6 with
$6.low \geq 5.dis$

1 is a cut vertex because
1 is root and has children $\{2,5\}$.

Computing low during DFS

Initial value of $u.low = u.dis$
- when DFSVisit(u) is entered.



$$u.low = \min\left\{ \min_{1 \le i \le k} \{c_i.low\},\ \min_{(u,v) \in E \text{ back edges}} \{v.dis\} \right\}$$

Total $RT = O(|V| + |E|)$ – linear.

LP1 Algorithms

1. Power: x^n

```
Power(x, n):
  base cases { if n=0 → return 1
               n=1 → return x
  else
      s ← Power(x, n/2)
      if n is even then
          return s * s
      else return s*s*x
```

```
s ← Power(x*x, n/2)
n even? return s
else    return s*x
```

$RT = O(\log n)$.

In LP1: Power(Num x, log n)

* ← product
\+ ← add

(a) Implement divide by 2

Power(Num x, Num n):

$n = \{a_0, a_1, \ldots a_d\}$ Base ~~B~~

$= a_0 + a_1 B + a_2 B^2 + \cdots + a_d B^d$

Right shift operation

$shift(n) = s = \{a_1, a_2, \ldots, a_d\}$ $\quad n = s * Base + a_0$

$x^n = (x^s)^{Base} * x^{a_0}$

2. Division, Mod, Squareroot

Mod: $a \% b \equiv a - (a/b) * b$

Be careful with negative numbers.

/ , $\sqrt{}$: use binary search.

a/b: Find x : $x * b \le a < (x+1) * b$

Handle sign separately; handle edge cases first.

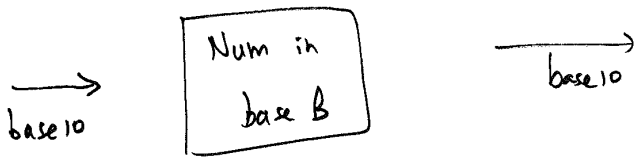
$\sqrt{a}$: x : $x * x \le a < (x+1)(x+1)$

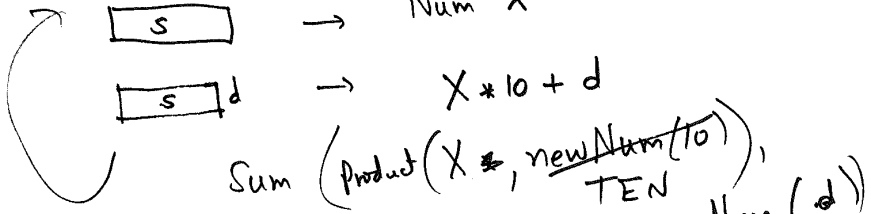
if $b > 1$. Answer $[1, a]$

---

3. I/O:

use StringBuilder class.





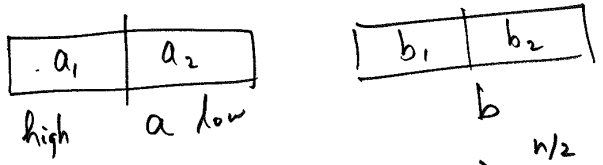
Multiplication operation (Product):

1. Long multiplication algorithm: create array to multiply in

↑ single static array for class will do

2. Divide & conquer algorithm (Karatsuba's algorithm):

multiply 2 n-bit numbers (n = power of 2)



Answer: $a_1 b_1 2^n + (a_1 b_2 + a_2 b_1) 2^{n/2} + a_2 b_2$

$\equiv a_1 b_1 2^n + [(a_1 + a_2)(b_1 + b_2) - a_1 b_1 - a_2 b_2] 2^{n/2} + a_2 b_2$

Product, shift; Product (sum, sum); subtract, subtract, shift; Product

Recurrence: $T(n) = 3T(n/2) + O(n)$

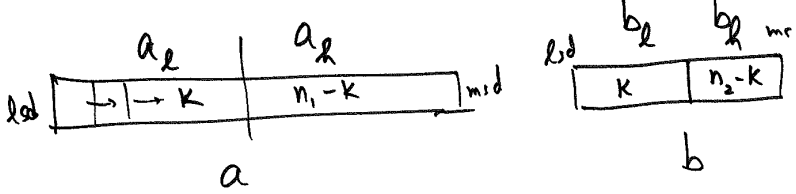
$T(n) = O(n^{\log_2 3})$

Karatsuba on LPI:

Product(a, b) # of digits of a ≥ # of digit of b

a's list has n_1 elements $n_1 \geq n_2$

b's list has n_2 elements



Let $k = n_2/2$

Result: $(a_h * b_h) * Base^{2k} + [(a_l + a_h) * (b_l + b_h) - a_h b_h - a_l b_l] * Base^{k} + a_l * b_l$

($a_h * b_h$): Recursive call to Product; $Base^{2k}$: shift; $Base^{k}$: shift

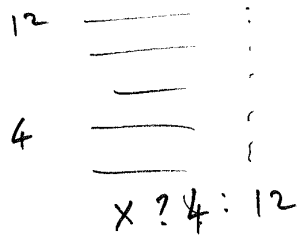
---

Level 1 – 3 : 100 points

Excellence credits: Level 3, 4 + quality

Level 3: Post~~order~~fix expression evaluation – use a stack for Num.

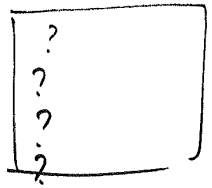
Level 4 (LPI) plan



Read input lines and store them in suitable class.

↳ convert infix expressions to postfix.

Array[ ]



goto 4

Create a hash table mapping labels with line numbers.