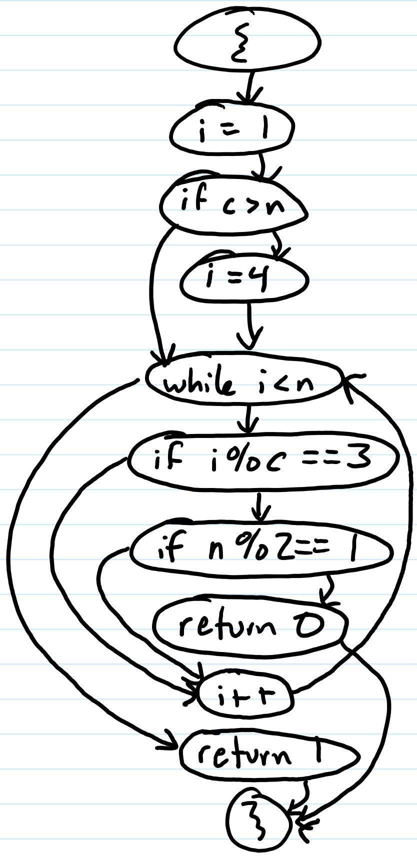
Control Flow Graphs

```
int foo(int n, int c)
{
   int i = 1;
   if (c > n)
   {
      i = 4;
   }
   while (i < n)
   {
      if (i % c == 3)
      {
         if (n % 2 == 1)
         {
            return 0;
         }
      }
      i++;
   }
   return 1;
}
```

c = 4   n = 5



vertex = line of code

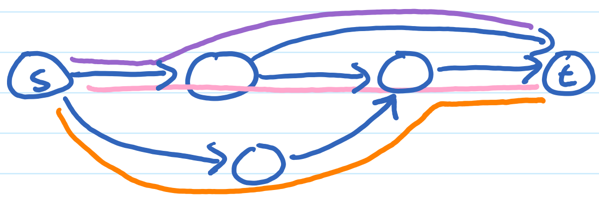
edge (u,v) means
v can follow u

paths { ⤳ return 0

shortest : 6   polynomial (BFS)
longest : 7   NP-complete
average : 6.5   ???

Average Path Length

AVERAGE PATH LENGTH: Given G, s, t, k, determine if the average length of all simple paths $s \rightsquigarrow t$ is at least k



average length = $2\frac{2}{3}$

A-P-L$(G, s, t, 2)$ = YES
A-P-L$(G, s, t, 4)$ = NO

AVERAGE-PATH-LENGTH $\overset{?}{\in}$ NP   no???

APL-BRUTE-FORCE(G, s, t, k)
tot, count ← 0
for each simple path p from $s \rightsquigarrow t$
    tot ← tot + len(p)
    count ← count + 1
return tot / count ≥ k

polynomial space

possibly $(n-2)!$ iterations
(more than polynomial time)

SAT-BRUTE-FORCE(φ)
$n \leftarrow$ # variables in φ
for $i := 0$ to $2^n - 1$ $\quad$ 2^n iterations
  generate i^{th} possible assignment A
  if A makes φ true
    return YES
return NO

polynomial space

TSP-BRUTE-FORCE(G, k)
for $i := 0$ to $n! - 1$ $\quad$ $n!$ iterations
  generate i^{th} permutation of vertices P
  $t \leftarrow$ total weight of corresponding tour
  if $t \le k$
    return YES
return NO

P = set of decision problems solvable in polynomial time

PSPACE = set of decision problems solvable in polynomial space

AVERAGE-PATH-LENGTH, SAT, TSP $\in$ PSPACE

P, NP, and PSPACE

$P \subseteq NP$

$P \subseteq PSPACE$ (constant space per step, so polynomial time → polynomial space)

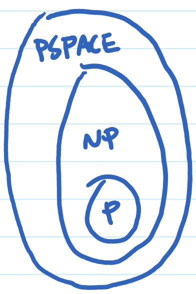
$NP \subseteq PSPACE$

$SAT \in PSPACE$

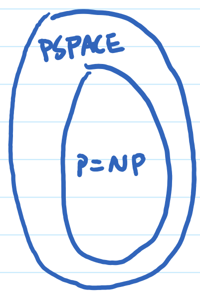
Let $X \in NP$. Then $X \le_p SAT$ (SAT is NP-complete)

and $X \in PSPACE$ (use the polynomial-time reduction)

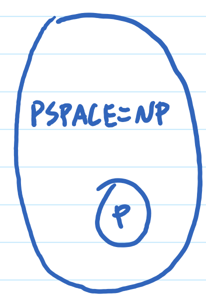
$P \subsetneq NP \subsetneq PSPACE$



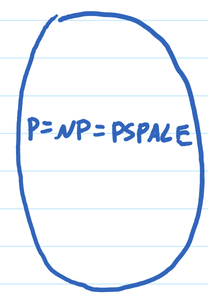
$P = NP \subsetneq PSPACE$



$P \subsetneq NP = PSPACE$



$P = NP = PSPACE$



PSPACE-complete

Problem X is PSPACE-complete if : 1) $X \in$ PSPACE
2) for all $Y \in$ PSPACE, $Y \leq_p X$

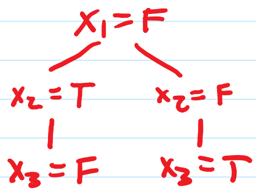
To prove P = PSPACE (and hence P = NP too) : find polynomial-time solution to some PSPACE-complete problem X

To prove P $\neq$ PSPACE : prove superpolynomial lower bound for some $X \in$ PSPACE
(says nothing about $P \stackrel{?}{=} NP$)

QSAT

3-CNF with odd # of variables

QSAT: Given φ, determine whether $\exists x_1 \forall x_2 \exists x_3 \ldots \varphi(x_1, \ldots, x_n)$ is true.



$(x_1 \vee x_2 \vee x_3) \wedge (\sim x_1 \vee x_2 \vee x_3) \wedge (\sim x_1 \vee \sim x_2 \vee x_3)$

leaves $\approx 2^{\frac{n}{2}}$ (2 branches every other level)
so not a polynomial-sized certificate

$(x_1 \vee x_2 \vee x_3) \wedge (\sim x_1 \vee x_2 \vee \sim x_3) \wedge (\sim x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee \sim x_3)$

for any x_1, pick $x_2 = F$ to make formula F no matter what x_3 is

QSAT $\in$ EXPTIME:
QSAT $\in$ PSPACE:

QSAT(φ)
return EVAL(φ, 1)

1 call per assignment
2^n assignments
so exponential time

EVAL(φ, i)
if i = # variables in φ (no vars left; just T and F)
  $r \leftarrow$ evaluate(φ)
  return r
$\varphi_T \leftarrow \varphi$ with $x_i = T$
$\varphi_F \leftarrow \varphi$ with $x_i = F$
$r_T \leftarrow$ EVAL(φ_T, i+1)
$r_F \leftarrow$ EVAL(φ_F, i+1)
if i % 2 = 1 (quantifier is $\exists$)
  return $r_T \vee r_F$
else (quantifier is $\forall$)
  return $r_T \vee r_F$

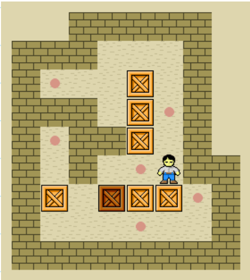
linear space per call
depth of recursion = n
polynomial space

PSPACE-complete (and EXPTIME)

QSAT ∈ PSPACE
QSAT ∈ EXPTIME → problems with $O(2^{p(n)})$ solutions for some polynomial $p(n)$

QSAT is PSPACE-complete (and so PSPACE ⊆ EXPTIME)
↳ reduce Y∈PSPACE to QSAT and solve by brute force

SOKOBAN ∈ EXPTIME
SOKOBAN ∉ P → P ≠ EXPTIME



P ⊆ NP ⊆ PSPACE ⊆ EXPTIME

at least one of these is proper
(we don't know which)

To show that X is PSPACE-complete, it suffices to 1) show X ∈ PSPACE
2) show QSAT $\leq_p$ X