

Stack ADT

Tiziana Ligorio
tligorio@hunter.cuny.edu

Today's Plan



Announcements

Stack ADT

Announcements and Syllabus Check

Please do not discourage others from asking questions
- refrain from unproductive comments/feedback

How was the Google info session yesterday???

Hunter Women in Computer Science:

- Facebook Group:

www.facebook.com/groups/HunterWomenInCS/

- Mailing List: info@hunterwics.com

- GitHub: <https://github.com/hunterwics>

Abstract Data Types

Bag

Set

Stack

Stack

34

A data structure representing a stack of things

Objects can be **pushed** onto the stack or **popped** from the stack

Stack

A data structure representing a stack of things

Objects can be **pushed** onto the stack or **popped** from the stack

34

Stack

127

34

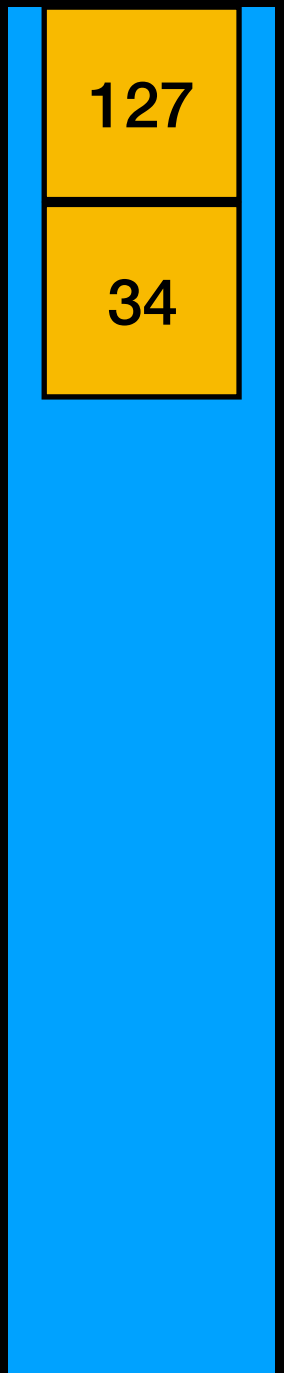
A data structure representing a stack of things

Objects can be **pushed** onto the stack or **popped** from the stack

Stack

A data structure representing a stack of things

Objects can be **pushed** onto the stack or **popped** from the stack



Stack

A data structure representing a stack of things

Objects can be **pushed** onto the stack or **popped** from the stack

13

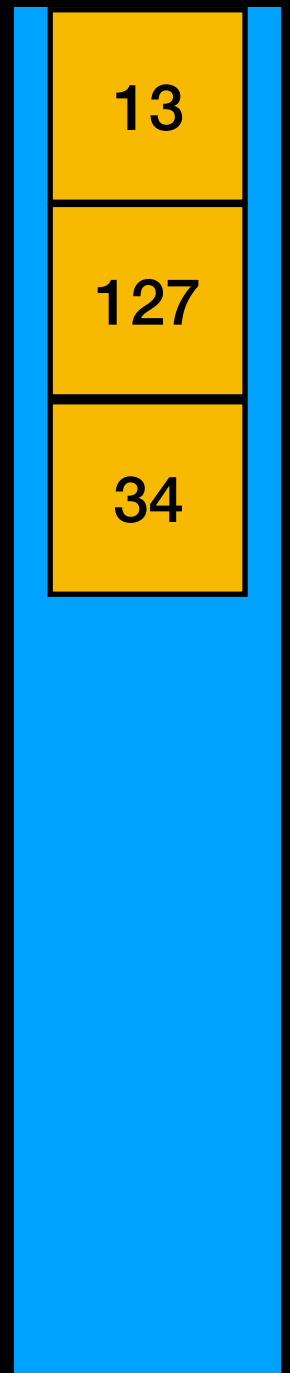
127

34

Stack

A data structure representing a stack of things

Objects can be **pushed** onto the stack or **popped** from the stack



Stack

13

A data structure representing a stack of things

Objects can be **pushed** onto the stack or **popped** from the stack

127

34

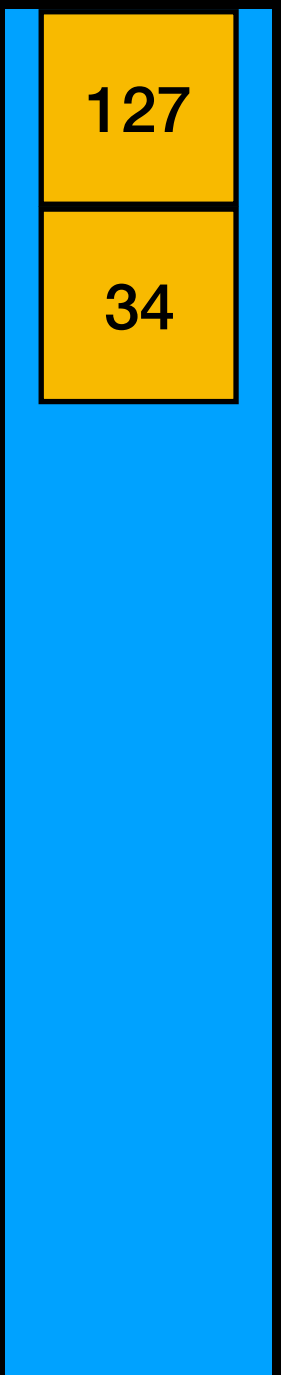
Stack

A data structure representing a stack of things

Objects can be **pushed** onto the stack or **popped** from the stack

LIFO: Last In First Out

Only top of stack is accessible (**top**), no other objects on the stack are visible



Applications

Very simple structure

Many applications:

- program stack

- balancing parenthesis

- evaluating postfix expressions

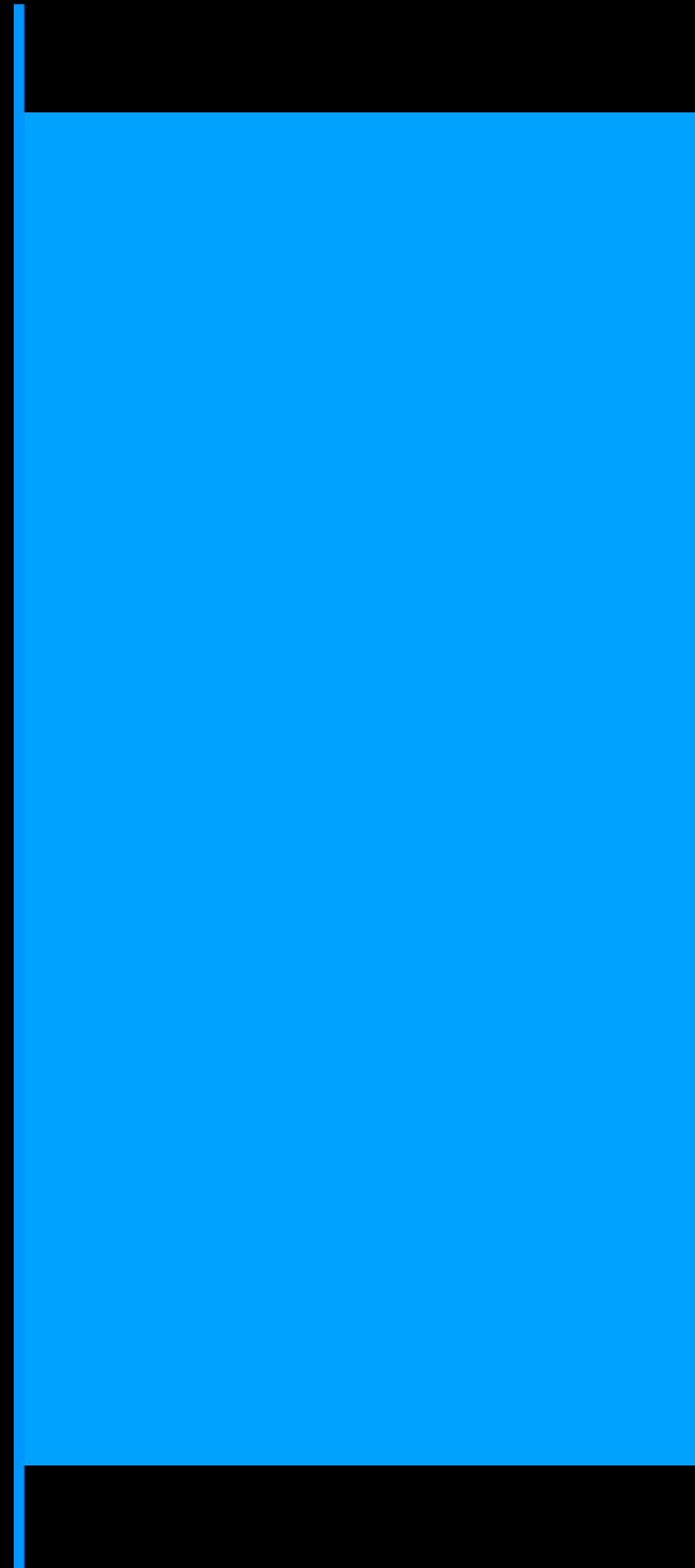
- backtracking

- ... and more

Program Stack

```
void f(int x, int y)
{
    int a;
    // stuff here
    if(a<13)
        a = g(a);
    // stuff here
}
```

```
int g(int z)
{
    int p ,q;
    // stuff here
    return q;
}
```

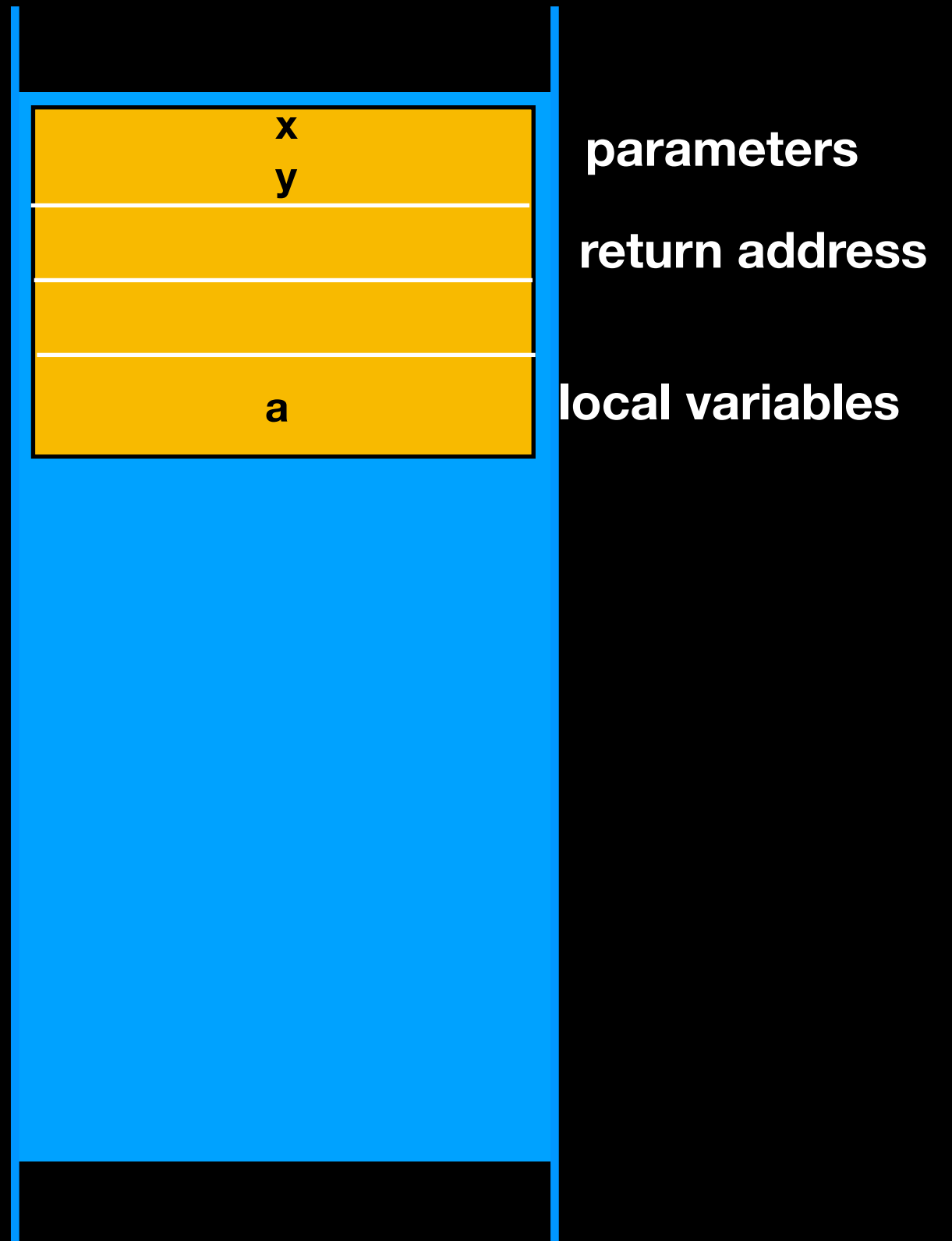


Program Stack

```
void f(int x, int y)
{
    int a;
    // stuff here
    if(a<13)
        a = g(a);
    // stuff here
}
```

```
int g(int z)
{
    int p ,q;
    // stuff here
    return q;
}
```

Stack Frame
for f



Program Stack

```
void f(int x, int y)
{
```

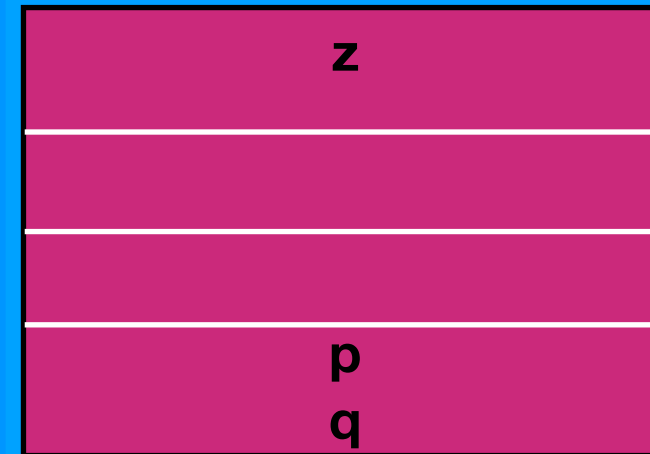
```
    int a;
    // stuff here
    if(a<13)
        a = g(a);
    // stuff here
}
```

```
int g(int z)
```

```
{
    int p ,q;
    // stuff here
    return q;
}
```

Stack Frame
for g

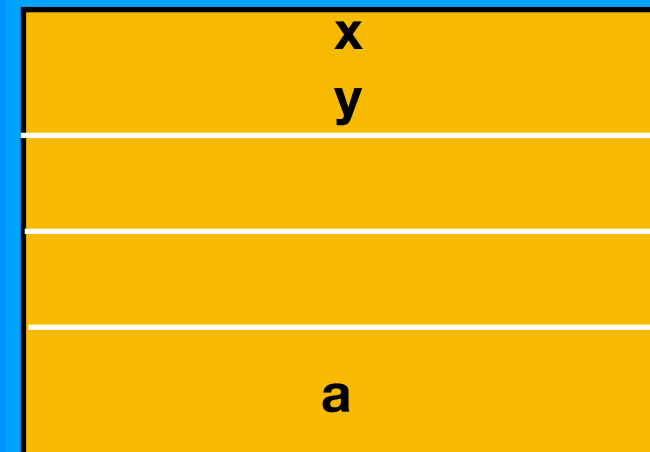
Stack Frame
for f



parameters

return address

local variables



parameters

return address

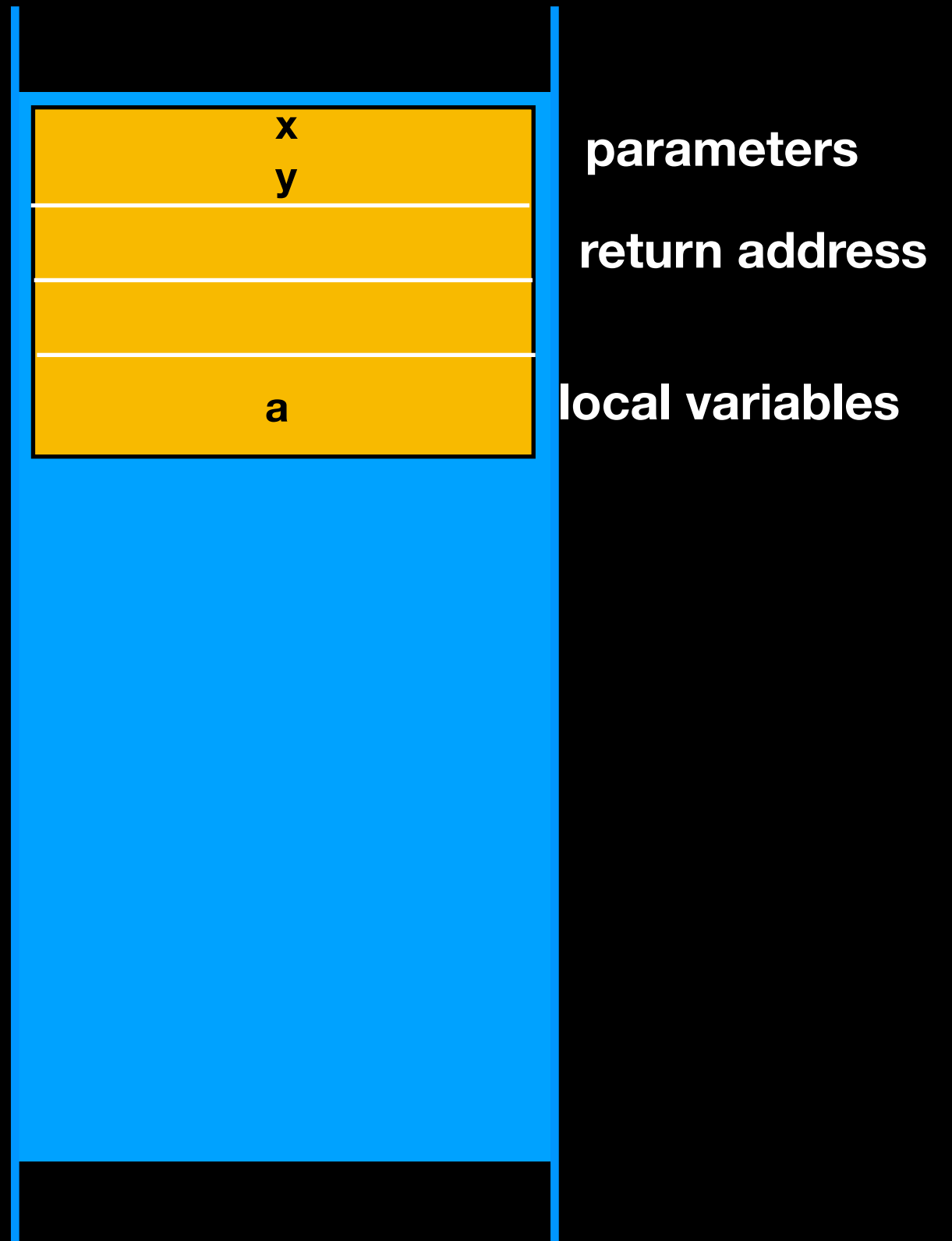
local variables

Program Stack

```
void f(int x, int y)
{
    int a;
    // stuff here
    if(a<13)
        a = g(a);
    // stuff here
}
```

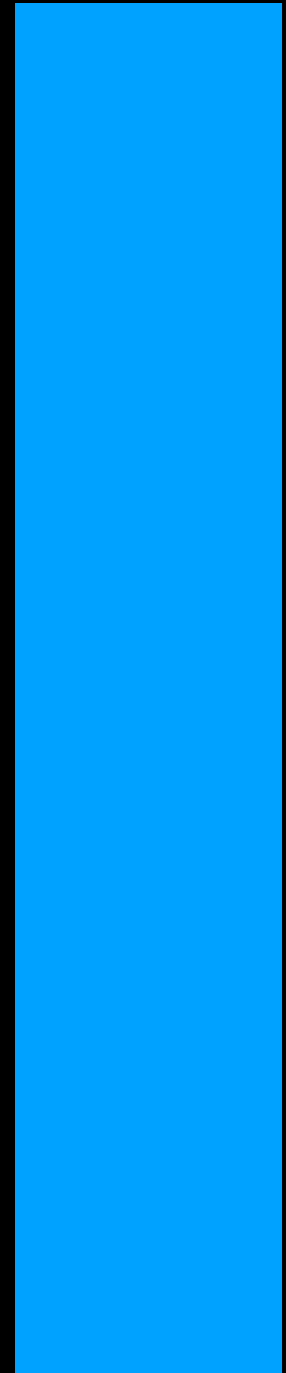

```
int g(int z)
{
    int p ,q;
    // stuff here
    return q;
}
```

Stack Frame
for f



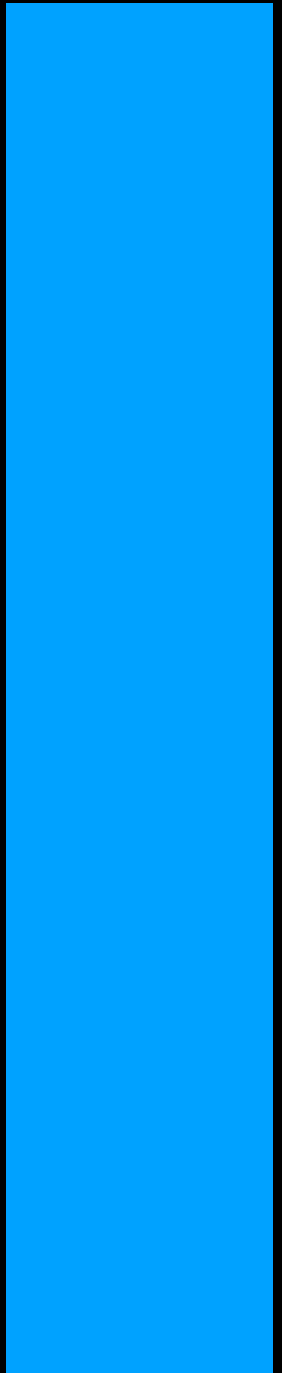

Balancing Parentheses

```
int f(){if(x*(y+z[i])<47){x += y}}
```



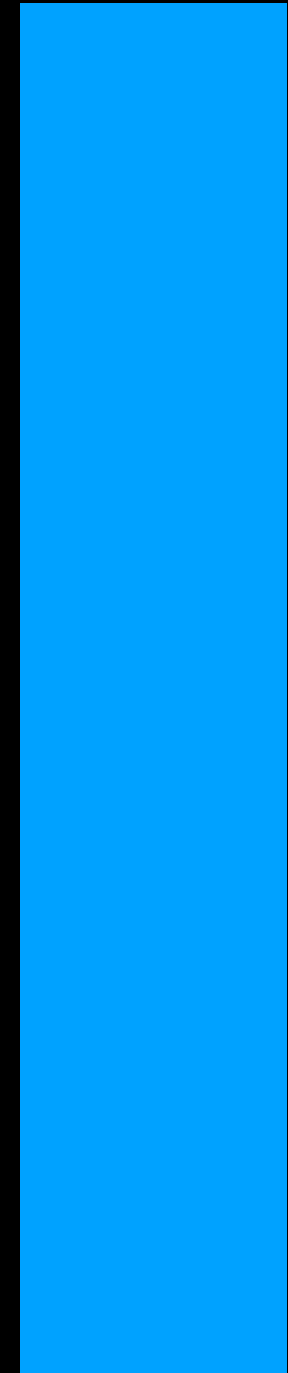

Balancing Parentheses

```
int f(){if(x*(y+z[i])<47){x += y}}
```



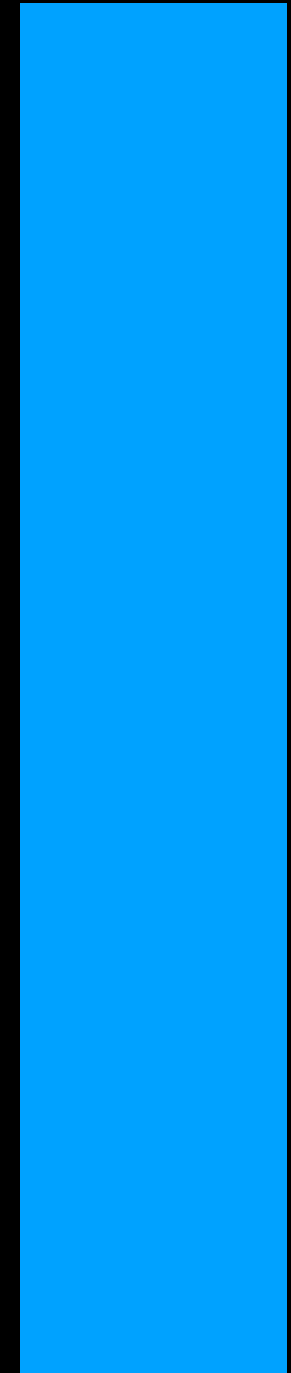

Balancing Parentheses

```
int f() {if(x*(y+z[i])<47) {x += y}}
```




Balancing Parentheses

```
int f() {if(x*(y+z[i])<47) {x += y}}
```



Balancing Parentheses

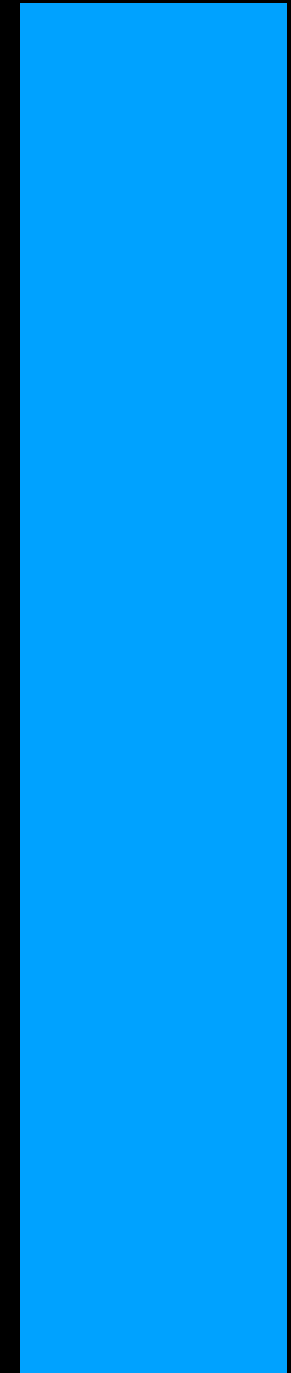

```
int f() {if(x*(y+z[i])<47){x += y}}
```



(


Balancing Parentheses

```
int f() {if(x*(y+z[i])<47){x += y}}
```




Balancing Parentheses

```
int f(){if(x*(y+z[i])<47){x += y}}
```



Balancing Parentheses


```
int f(){if(x*(y+z[i])<47){x += y}}
```



}

Balancing Parentheses


```
int f(){if(x*(y+z[i])<47){x += y}}
```



}

Balancing Parentheses

```
int f(){if(x*(y+z[i])<47){x += y}}
```



Balancing Parentheses

```
int f(){if(x*(y+z[i])<47){x += y}}
```

↑

(
{

Balancing Parentheses

```
int f(){if(x*(y+z[i])<47){x += y}}
```

↑

(
{

Balancing Parentheses

```
int f(){if(x*(y+z[i])<47){x += y}}
```



Balancing Parentheses

```
int f(){if(x*(y+z[i])<47){x += y}}
```



Balancing Parentheses

```
int f(){if(x*(y+z[i])<47){x += y}}
```



Balancing Parentheses

```
int f(){if(x*(y+z[i])<47){x += y}}
```



Balancing Parentheses

```
int f(){if(x*(y+z[i])<47){x += y}}
```



Balancing Parentheses

```
int f(){if(x*(y+z[i])<47){x += y}}
```




Balancing Parentheses

```
int f(){if(x*(y+z[i])<47){x += y}}
```



Balancing Parentheses


```
int f(){if(x*(y+z[i])<47){x += y}}
```



(
{

Balancing Parentheses

```
int f(){if(x*(y+z[i])<47){x += y}}
```



Balancing Parentheses


```
int f(){if(x*(y+z[i])<47){x += y}}
```



(
{

Balancing Parentheses

```
int f(){if(x*(y+z[i])<47){x += y}}
```

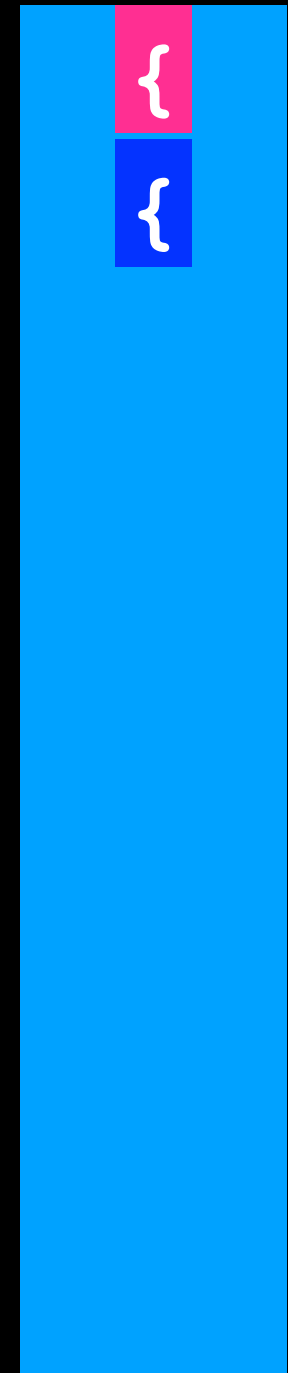


{

Balancing Parentheses

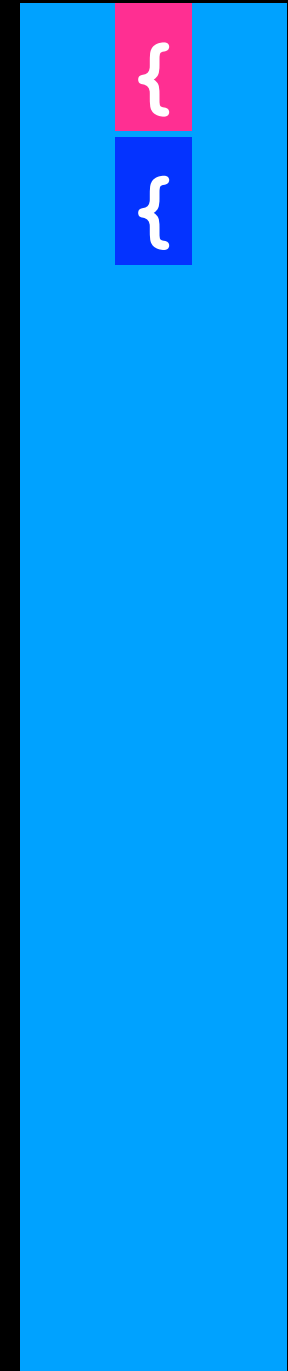
```
int f(){if(x*(y+z[i])<47){x += y}}
```

↑



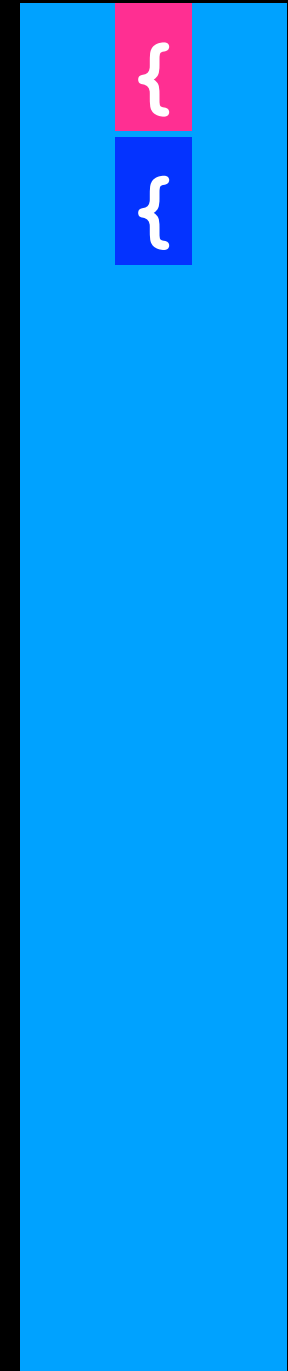
Balancing Parentheses

```
int f(){if(x*(y+z[i])<47){x += y}}
```



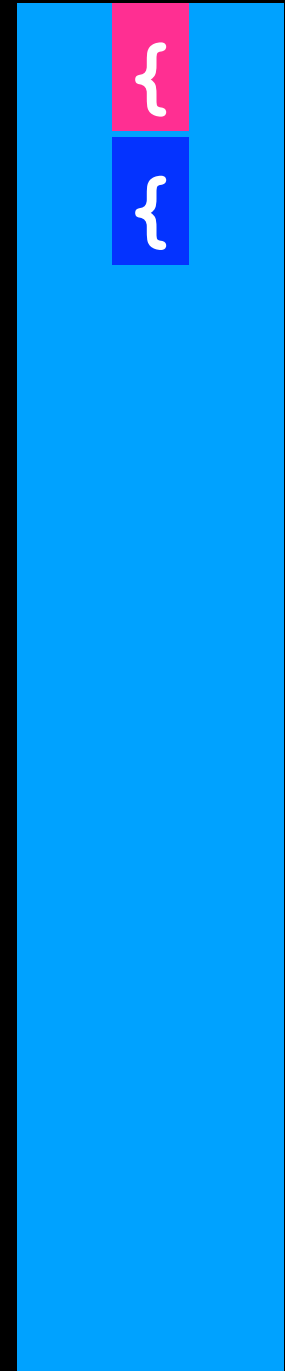
Balancing Parentheses

```
int f(){if(x*(y+z[i])<47){x += y}}
```



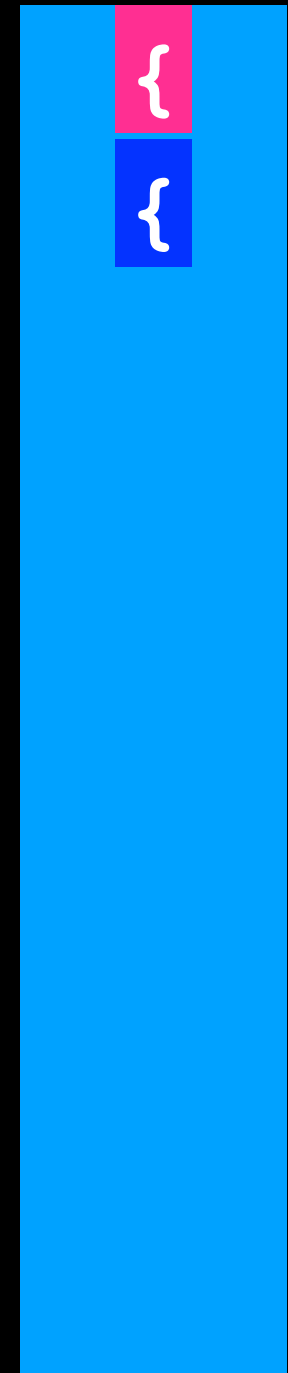
Balancing Parentheses

```
int f(){if(x*(y+z[i])<47){x += y}}
```




Balancing Parentheses

```
int f(){if(x*(y+z[i])<47){x += y}}
```



Balancing Parentheses

```
int f(){if(x*(y+z[i])<47){x += y}}
```



Balancing Parentheses

```
int f(){if(x*(y+z[i])<47){x += y}}
```



Finished reading
Stack is empty
Parentheses are balanced



Balancing Parentheses

```
int f(){if(x*(y+z[i])<47){x += y}
```



}

Finished reading
Stack not empty
Parentheses NOT balanced

Balancing Parentheses

```
for(char ch : st)
{
    if ch is an open parenthesis character
        push it on the stack
    else if ch is a close parenthesis character
        if it matches the top of the stack
            pop the stack
        else
            return unbalanced
    // else it is not a parenthesis
}

if stack is empty
    return balanced
else
    return unbalanced
```

Evaluating Postfix Expressions

Operator applies to the two operands immediately preceding it

Infix:	Postfix:
$2 * (3 + 4)$	$2\ 3\ 4\ +\ *$

Assumptions:

- string is syntactically correct postfix expression
- No unary operators
- No exponentiation operation
- Operands are single integer values

Evaluating Postfix Expressions

Postfix:

2 3 4 + *



2

Evaluating Postfix Expressions

Postfix:

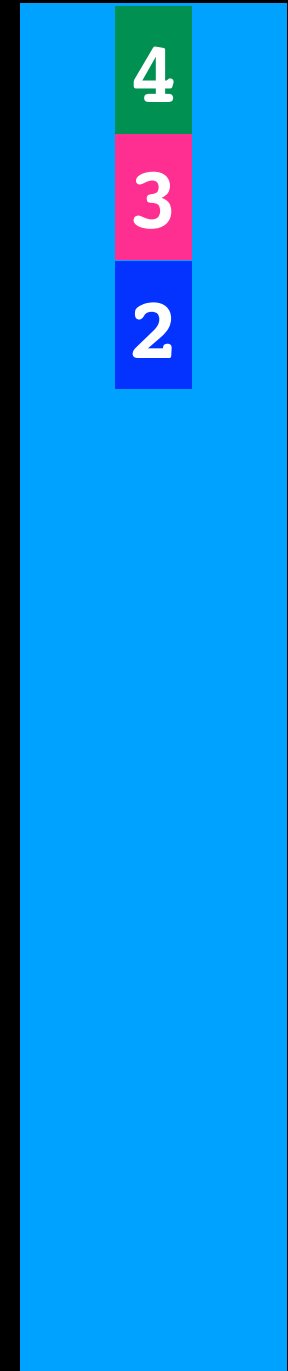

2 3 4 + *



Evaluating Postfix Expressions

Postfix:

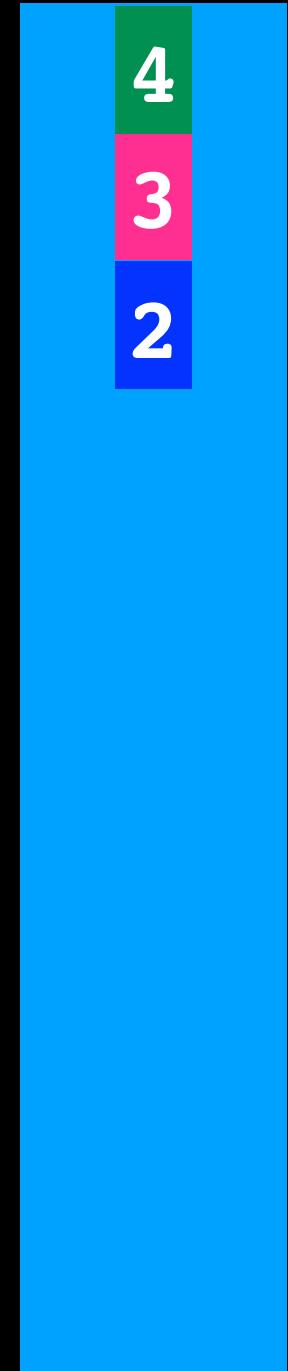

2 3 4 + *



Evaluating Postfix Expressions

Postfix:


2 3 4 + *



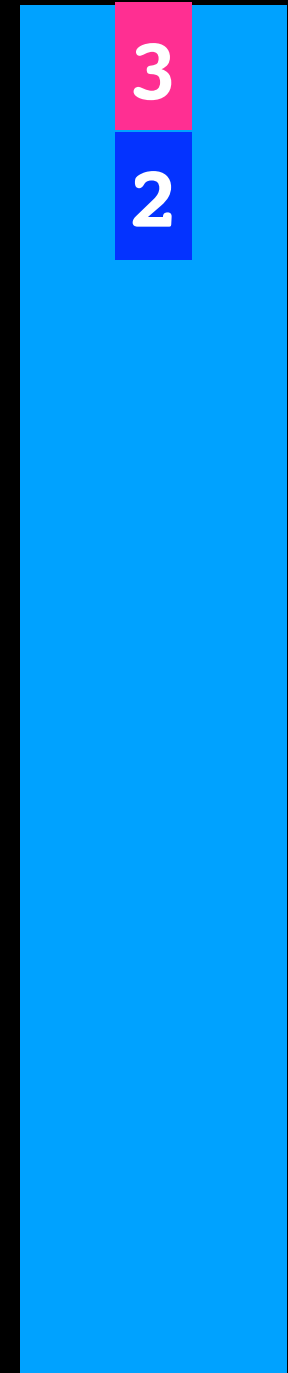
Evaluating Postfix Expressions

Postfix:

2 3 4 + *




4 +



Evaluating Postfix Expressions

Postfix:

2 3 4 + *



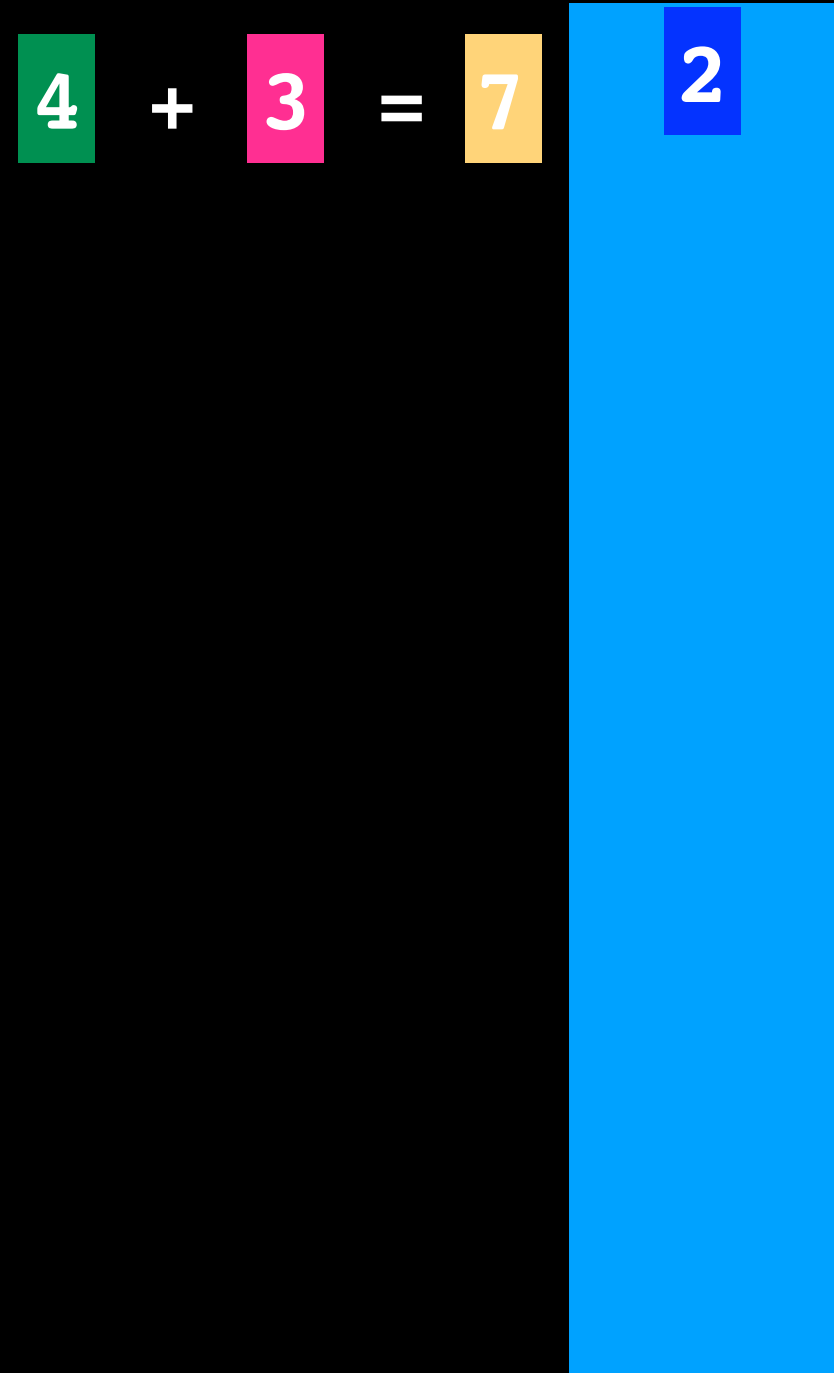

4 + 3

2

Evaluating Postfix Expressions

Postfix:


2 3 4 + *



Evaluating Postfix Expressions

Postfix:

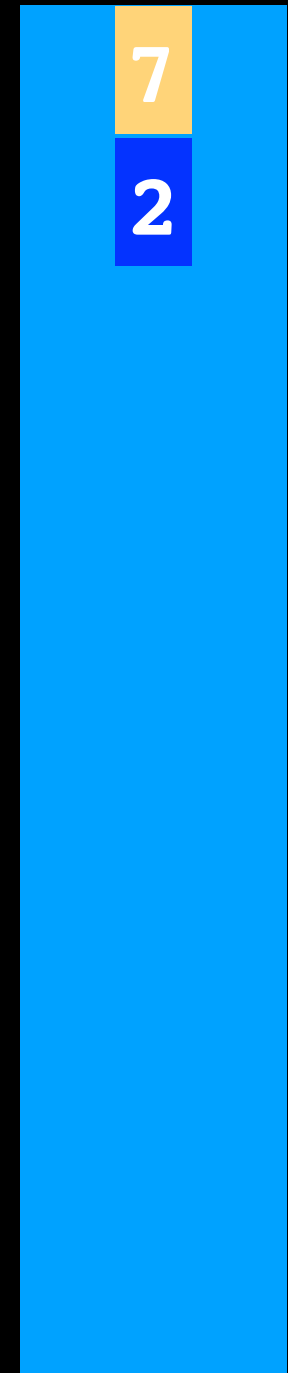
2 3 4 + *



Evaluating Postfix Expressions

Postfix:

2 3 4 + *
↑



Evaluating Postfix Expressions

Postfix:

2 3 4 + *
↑

7

2

Evaluating Postfix Expressions

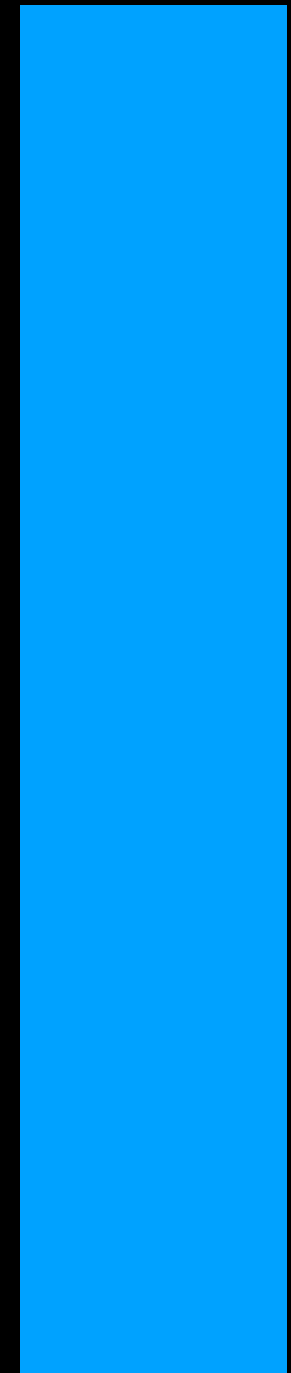
Postfix:

2 3 4 + *
↑

7

*

2

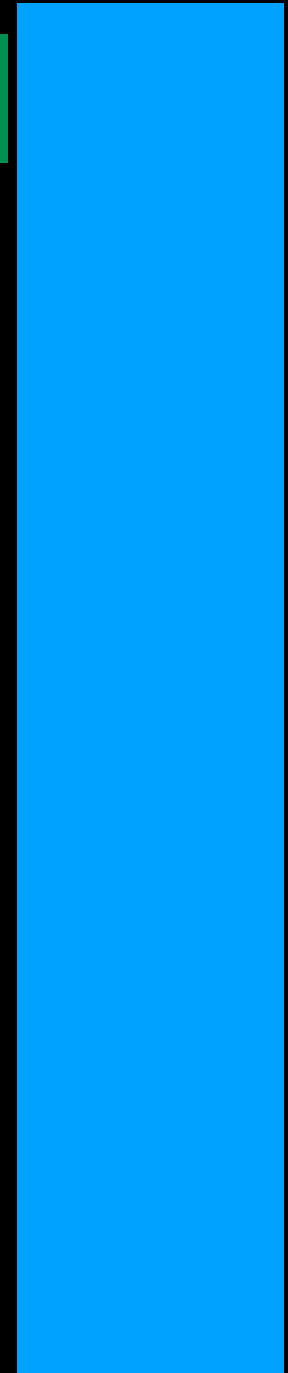


Evaluating Postfix Expressions

Postfix:

2 3 4 + *
↑

7 * 2 = 14



Evaluating Postfix Expressions

Postfix:

2 3 4 + *



Done reading string
The top of the stack
is the result

14



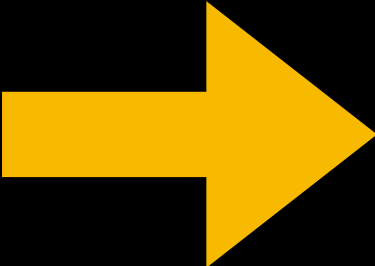
Evaluating Postfix Expressions

```
for(char ch : st)
{
    if ch is an operand
        push it on the stack
    else // ch is an operator op
    {
        //evaluate and push the result
        operand2 = pop stack
        operand1 = pop stack
        result = operand1 op operand2
        push result on stack
    }
}
```


In-Class Task

Draw a sequence of stack pushes and pops to convert the infix expression below into postfix:

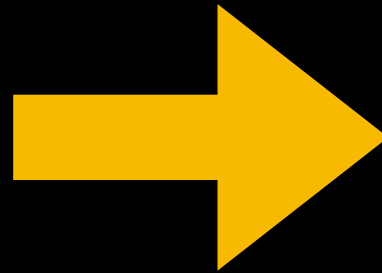
Infix:
 $2 * (3 + 4)$



Postfix:
 $2 3 4 + *$

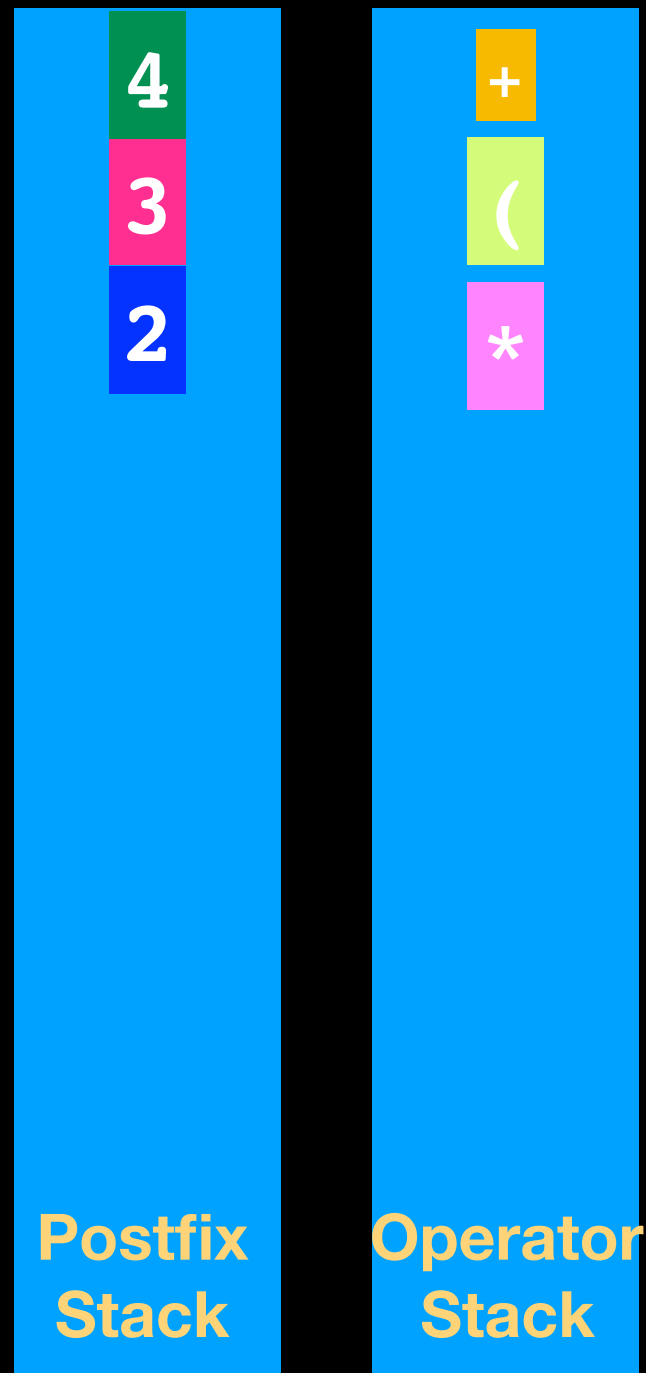
Tip: use 2 stacks, one for operators and one for the postfix expression

Infix:
 $2 * (3 + 4)$

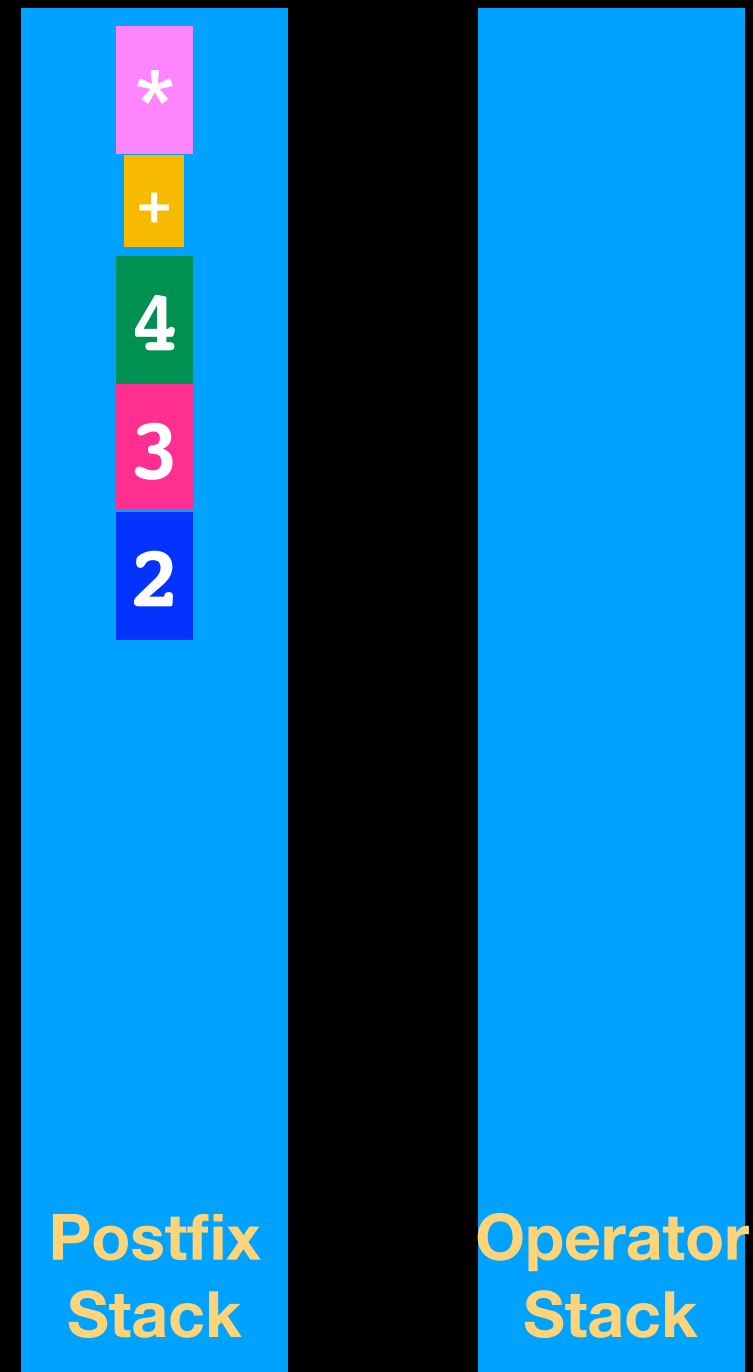


Postfix:
 $2 3 4 + *$

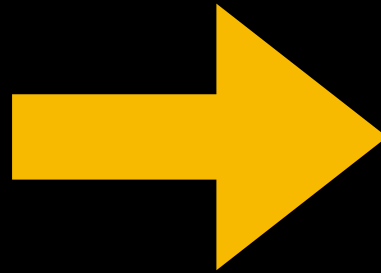
1. Read characters onto corresponding stack



2. Pop operator stack and push it on postfix stack ignoring '('



Infix:
 $(2 * 3) + 4$



Postfix:
 $2 3 * 4 +$

1. Read characters onto corr. stack

2. If reading a ')' ignore it, move operators to Postfix Stack until a '(' discard it and continue

3. Keep reading until ')' -> 2. or end of string -> 4.

4. Move operators to Postfix Stack

3
2

*
(

*
3
2

4
*
3
2

+

+
4
*
3
2

Postfix
Stack

Operator
Stack

Postfix
Stack

Operator
Stack

Postfix
Stack

Operator
Stack

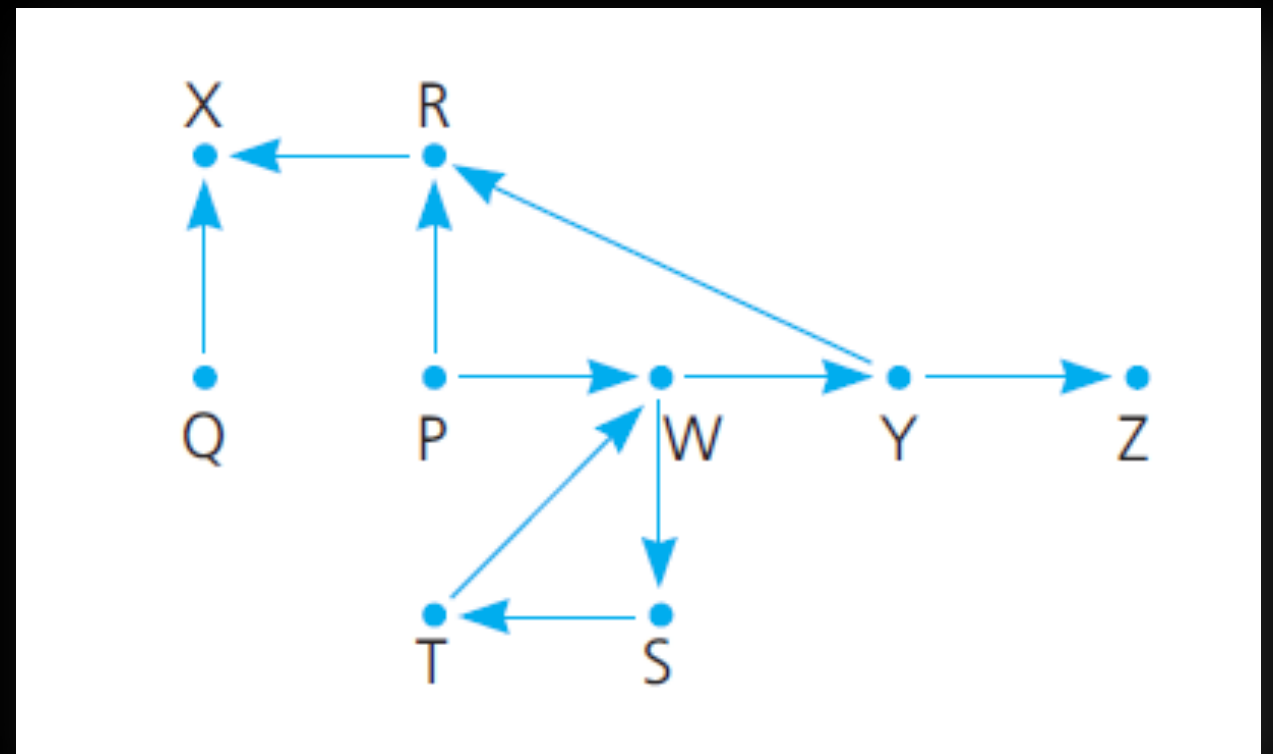
Postfix
Stack

Operator
Stack

Search a Flight Map

Fly from Origin to Destination following map

1. Reach destination
2. Reach city with no departing flights
3. Go in circles forever



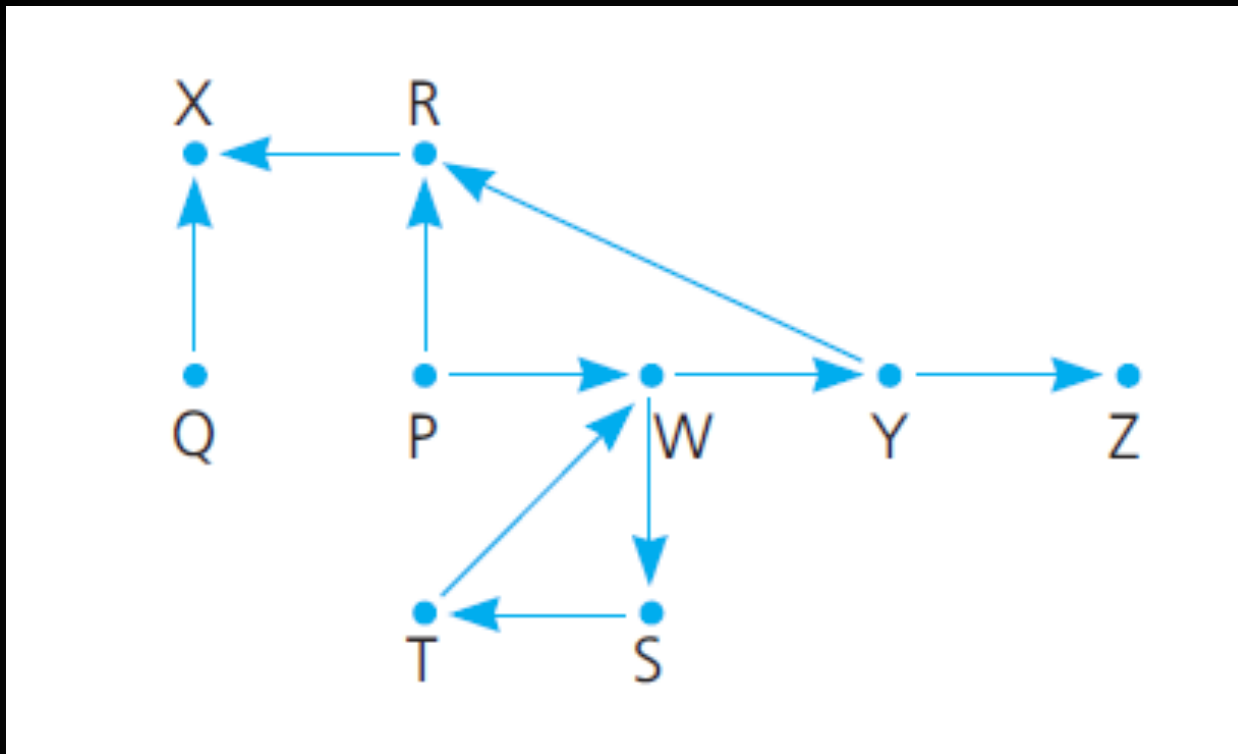
Backtracking

Avoid dead end by backtracking

Avoid traveling in circles by marking visited cities

Origin = P , **Destination = Z**

P

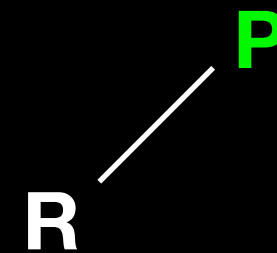
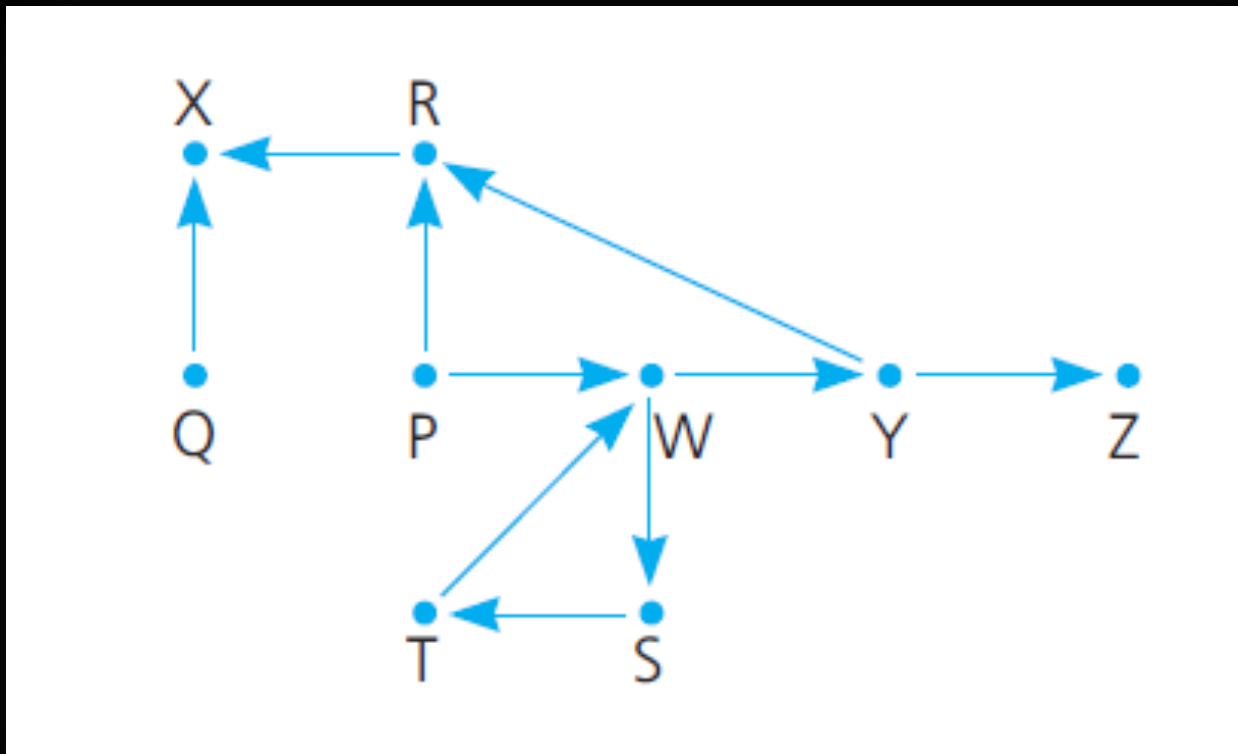


Backtracking

Avoid dead end by backtracking

Avoid traveling in circles by marking visited cities

Origin = P , **Destination = Z**

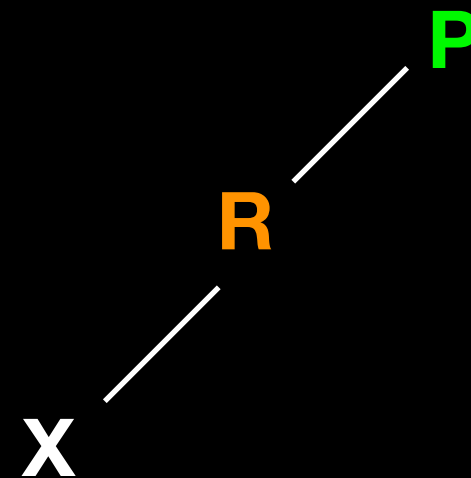
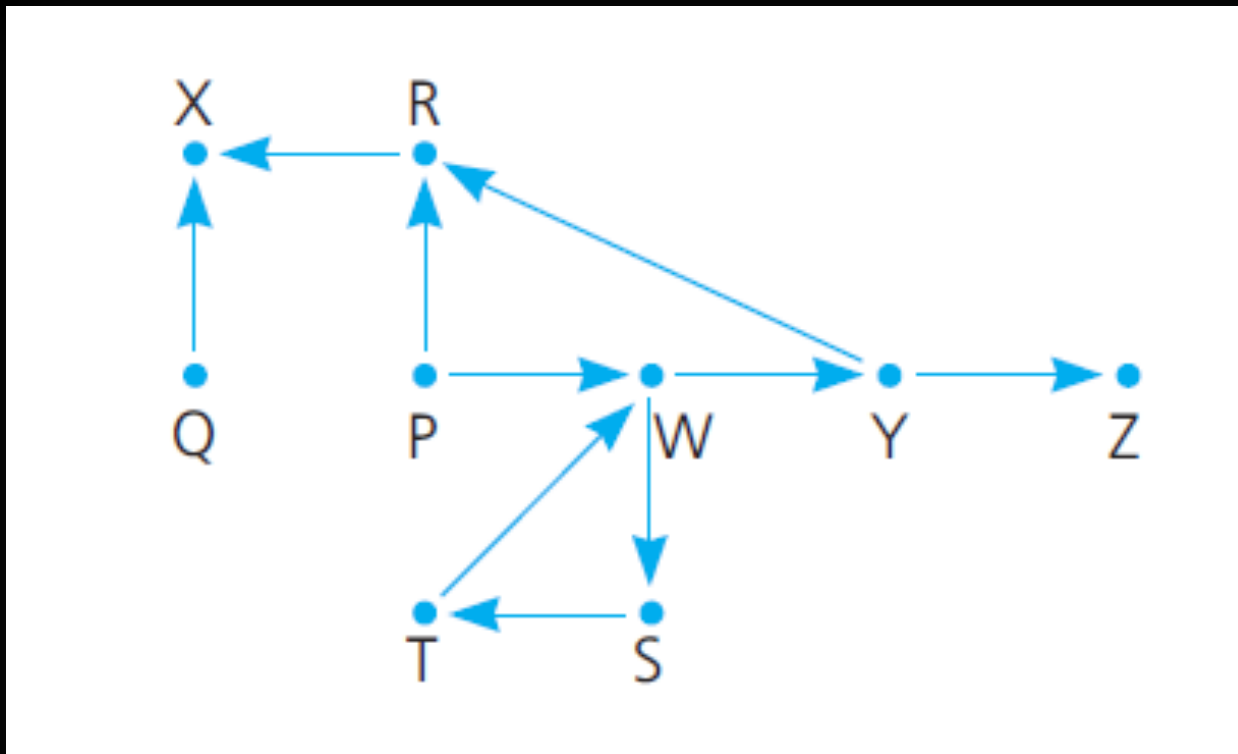


Backtracking

Avoid dead end by backtracking

Avoid traveling in circles by marking visited cities

Origin = P , **Destination = Z**

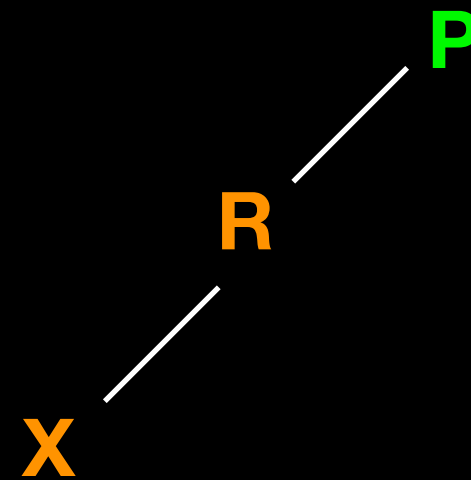
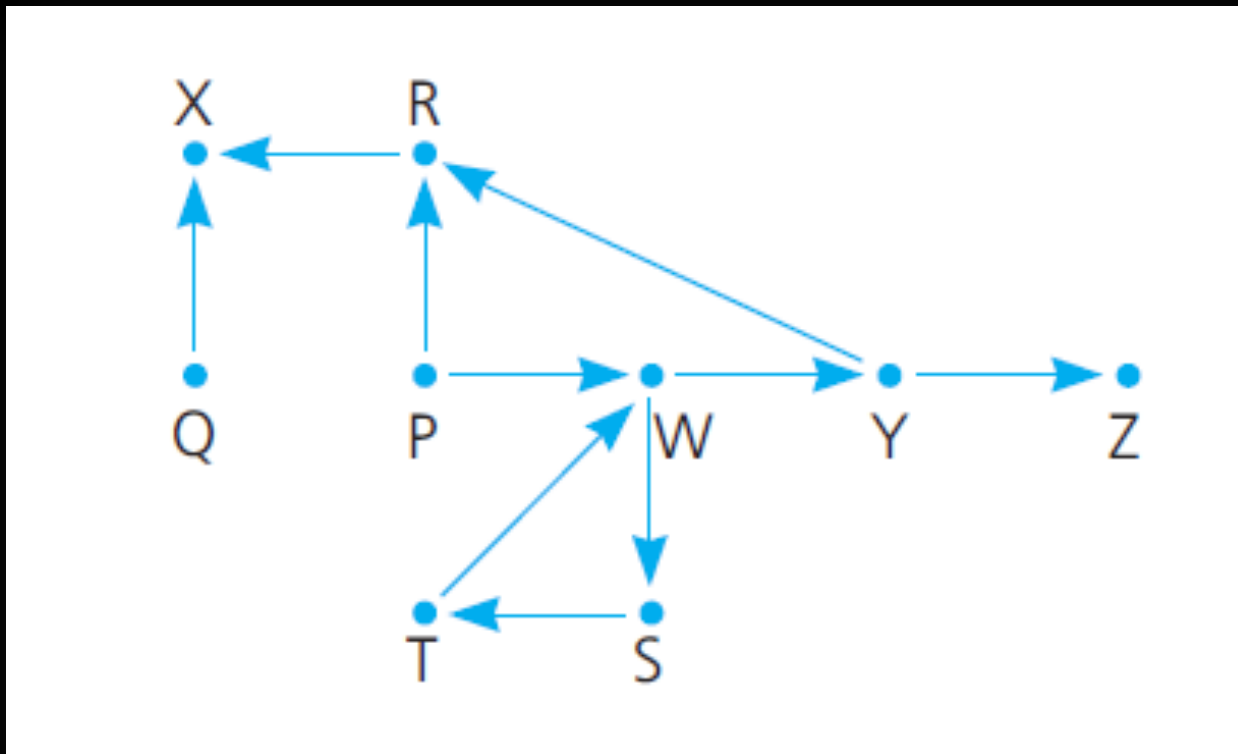


Backtracking

Avoid dead end by backtracking

Avoid traveling in circles by marking visited cities

Origin = P , **Destination = Z**

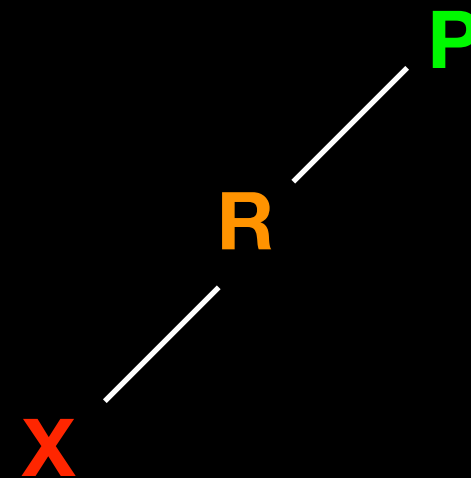
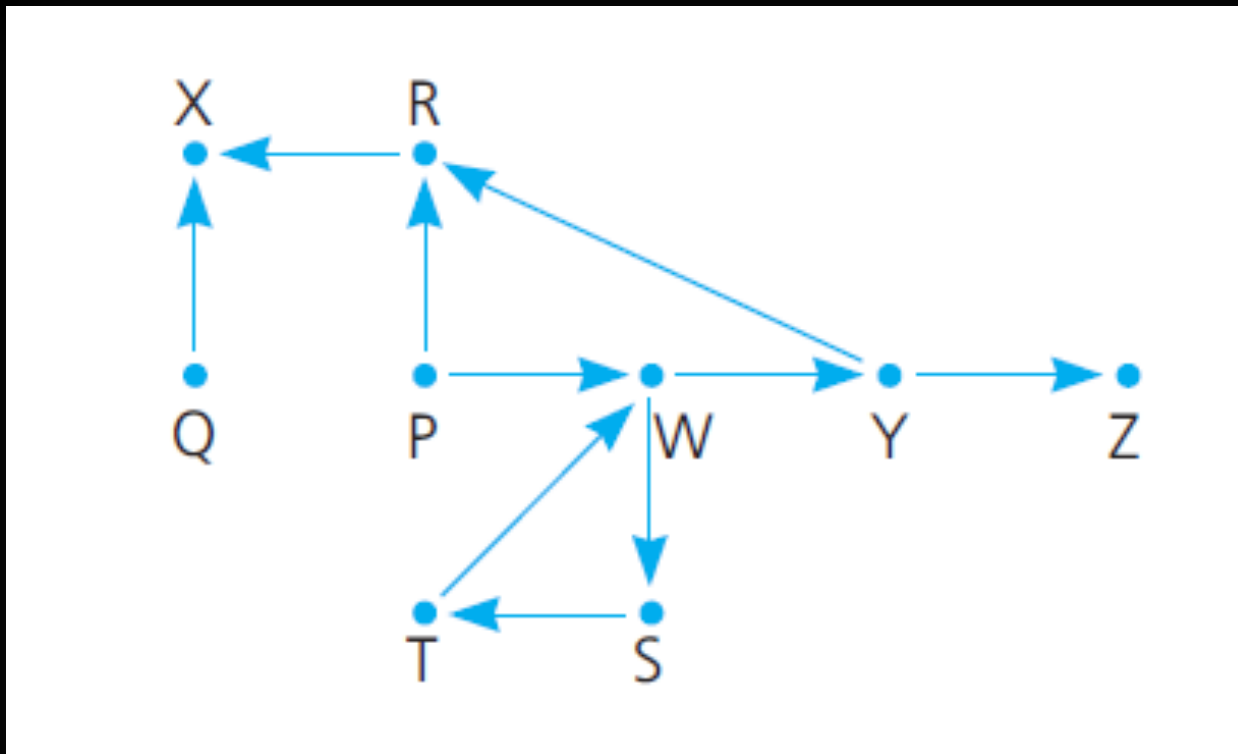


Backtracking

Avoid dead end by backtracking

Avoid traveling in circles by marking visited cities

Origin = P , **Destination = Z**

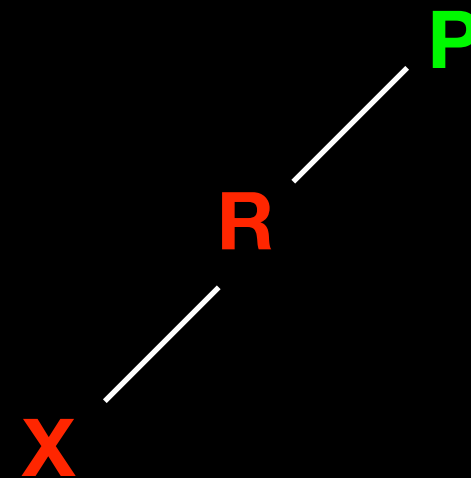
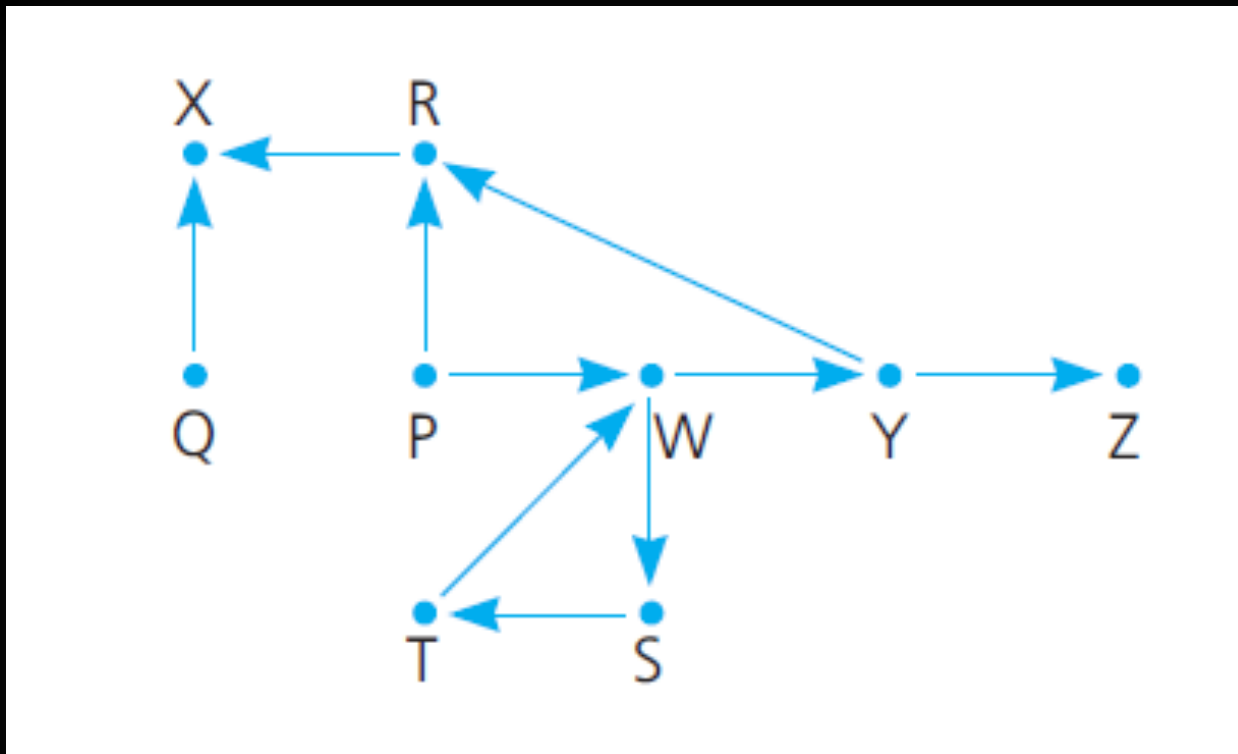


Backtracking

Avoid dead end by backtracking

Avoid traveling in circles by marking visited cities

Origin = P , **Destination = Z**

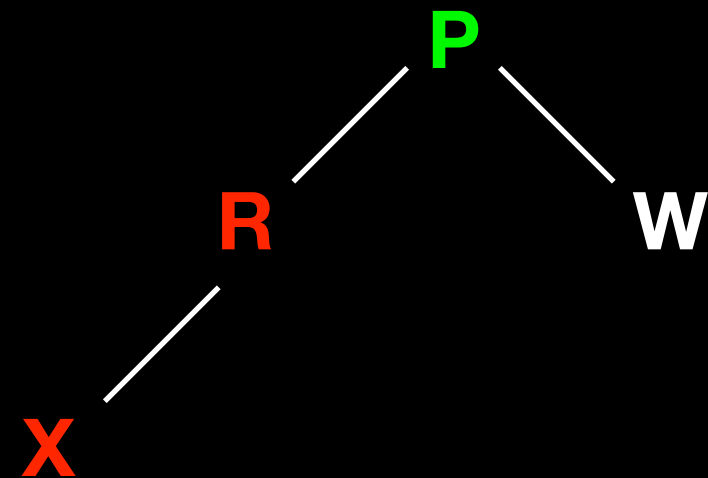
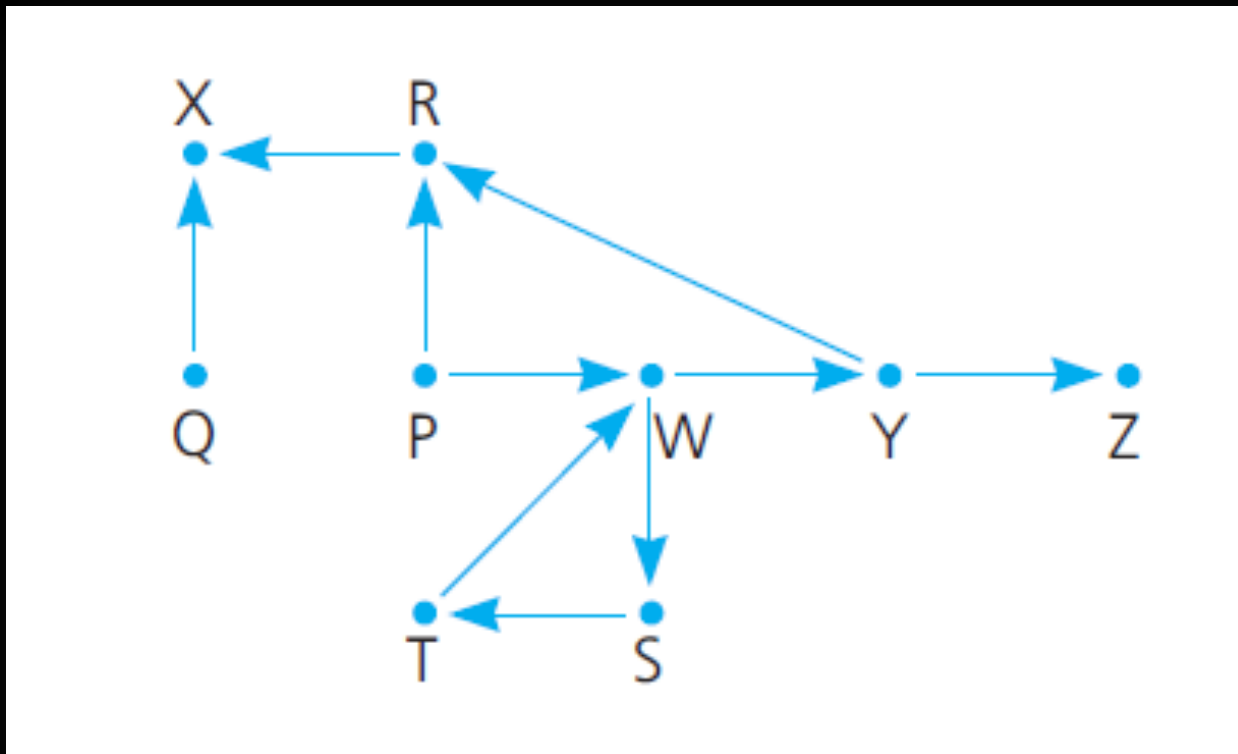


Backtracking

Avoid dead end by backtracking

Avoid traveling in circles by marking visited cities

Origin = P , **Destination = Z**

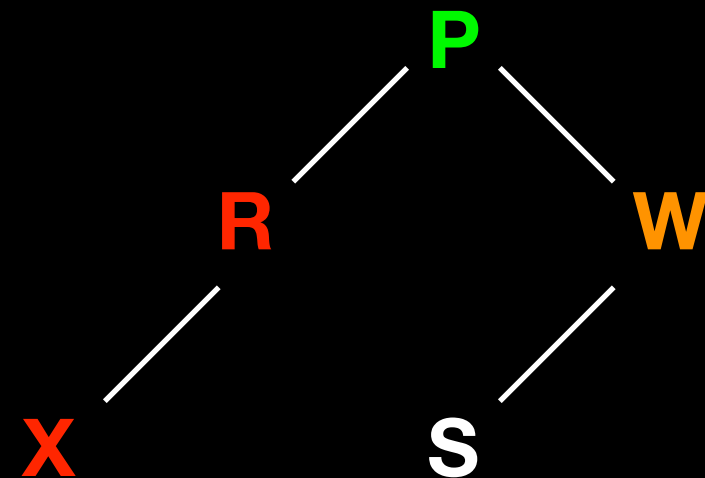
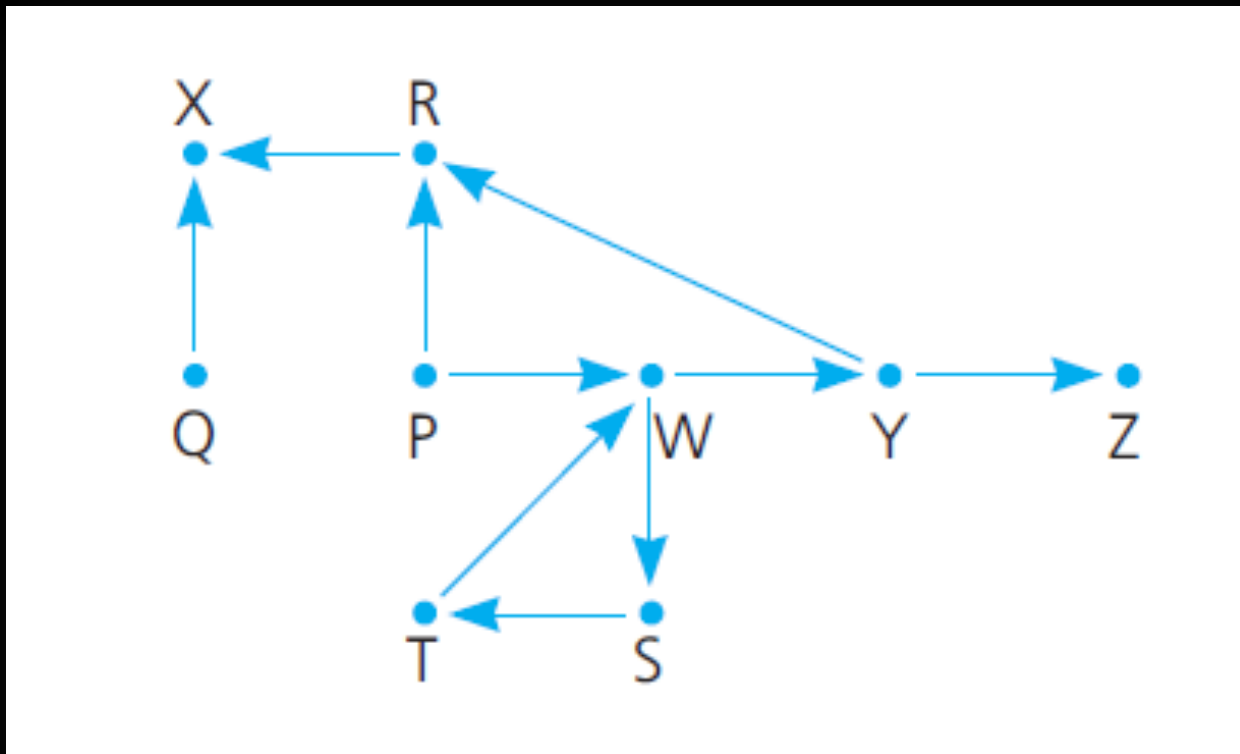


Backtracking

Avoid dead end by backtracking

Avoid traveling in circles by marking visited cities

Origin = P , **Destination = Z**

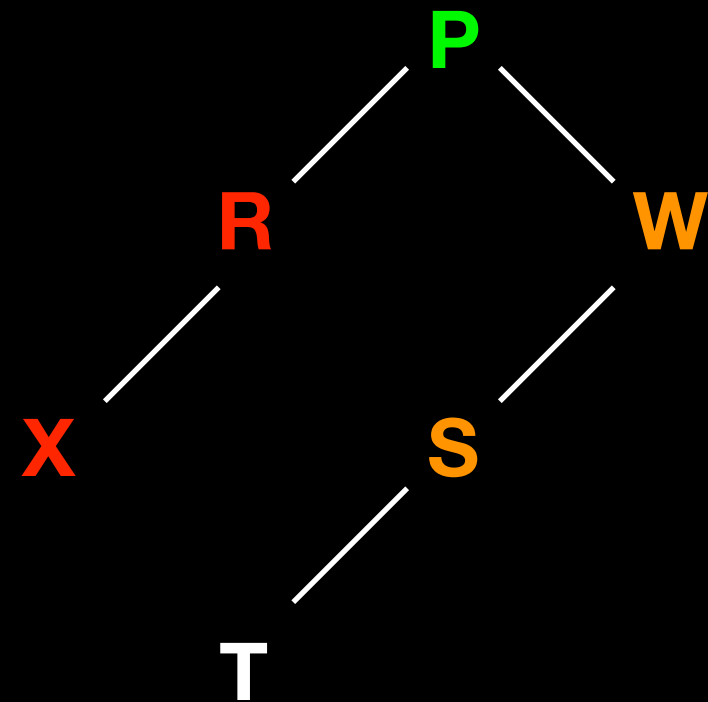
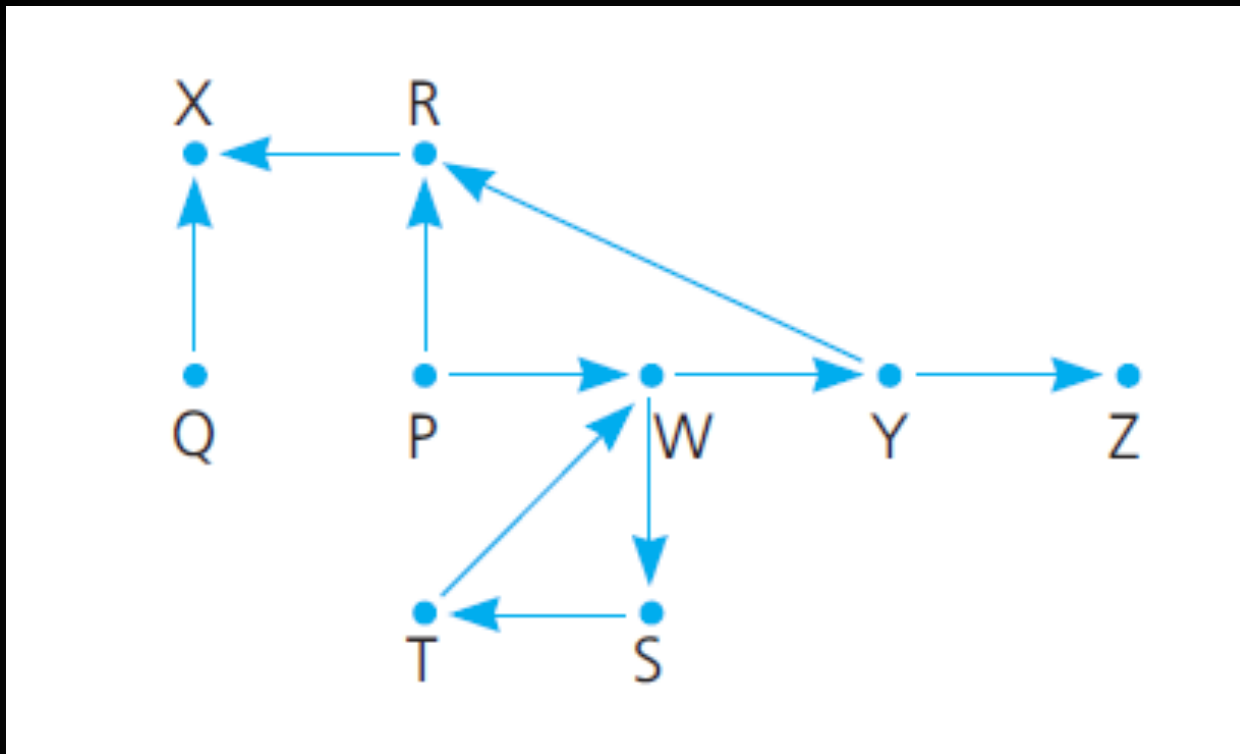


Backtracking

Avoid dead end by backtracking

Avoid traveling in circles by marking visited cities

Origin = P , Destination = Z

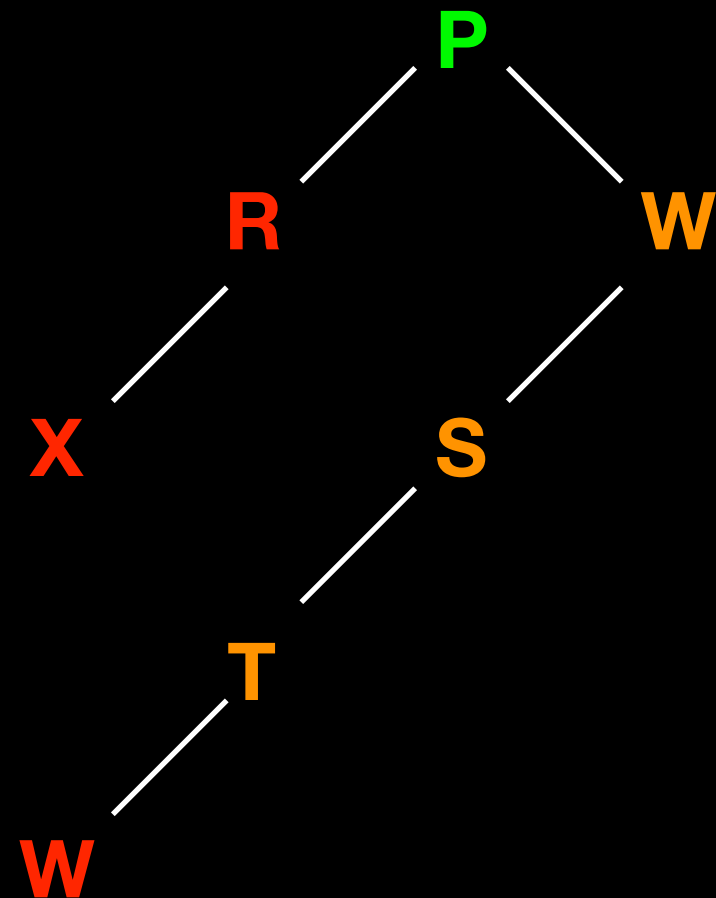
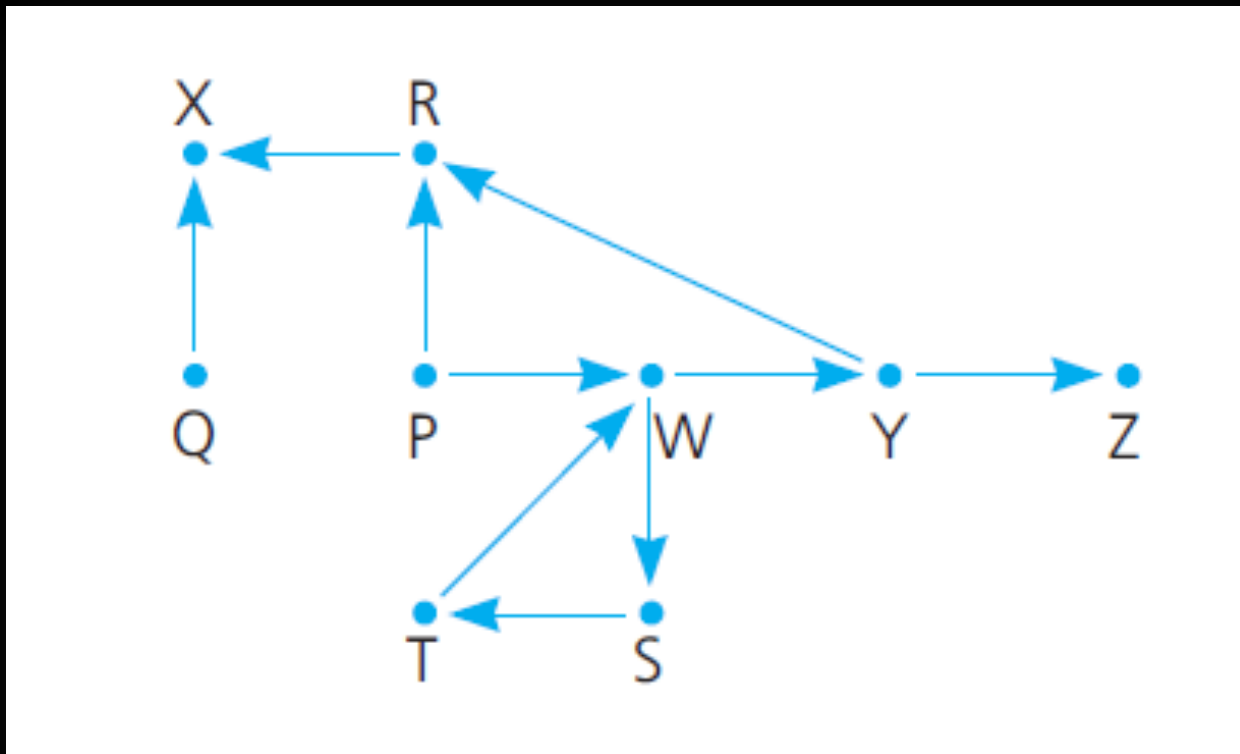


Backtracking

Avoid dead end by backtracking

Avoid traveling in circles by marking visited cities

Origin = P , **Destination = Z**

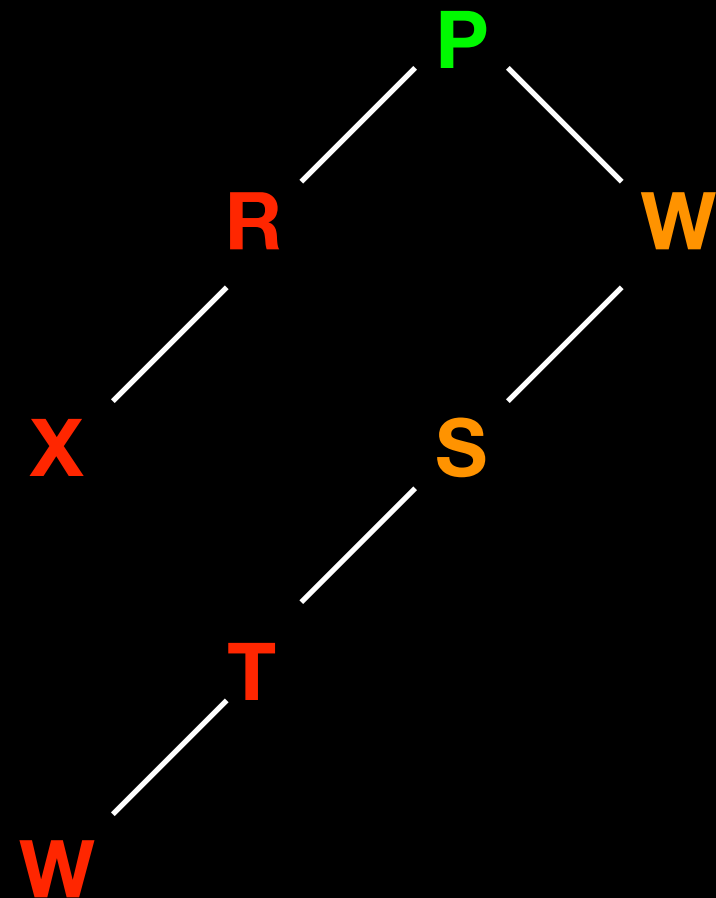
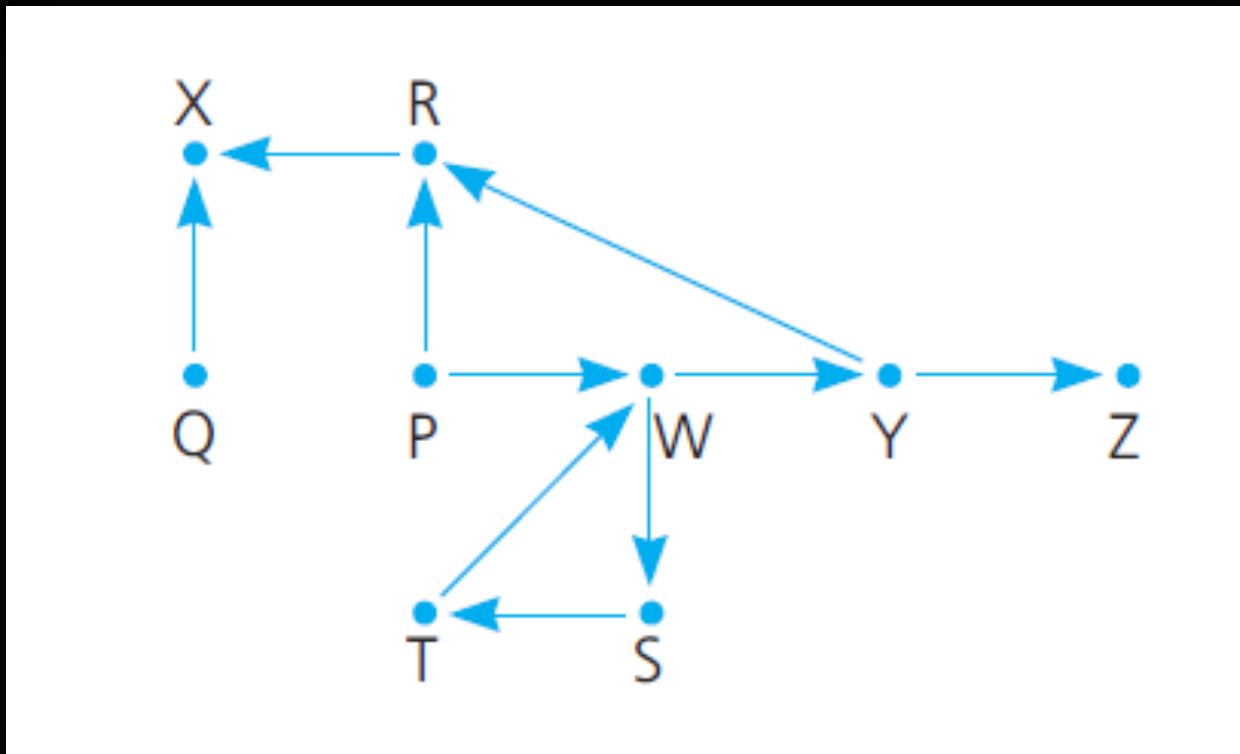


Backtracking

Avoid dead end by backtracking

Avoid traveling in circles by marking visited cities

Origin = P , **Destination = Z**

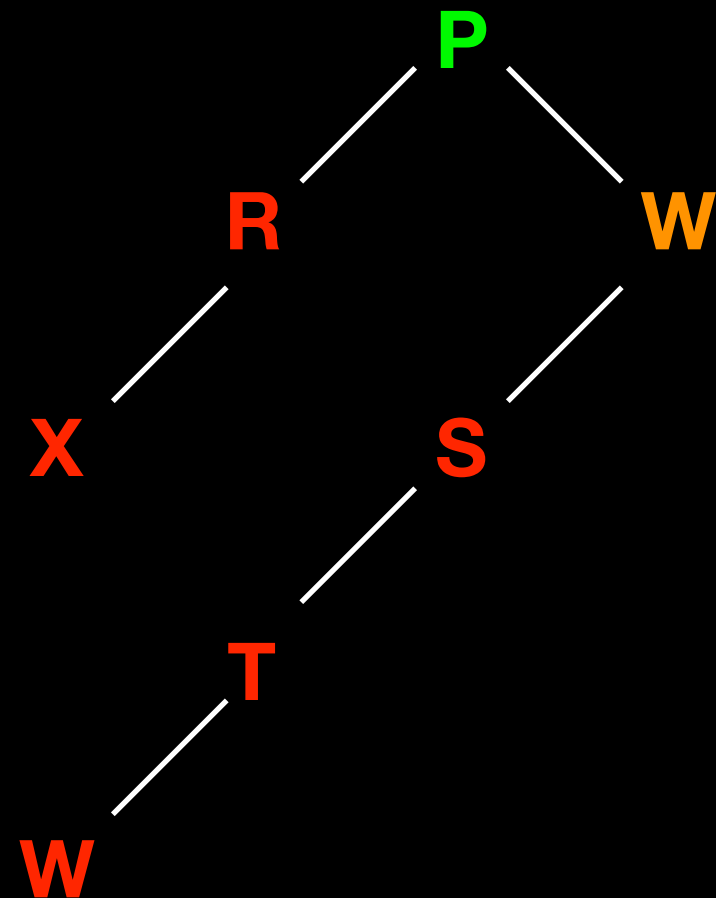
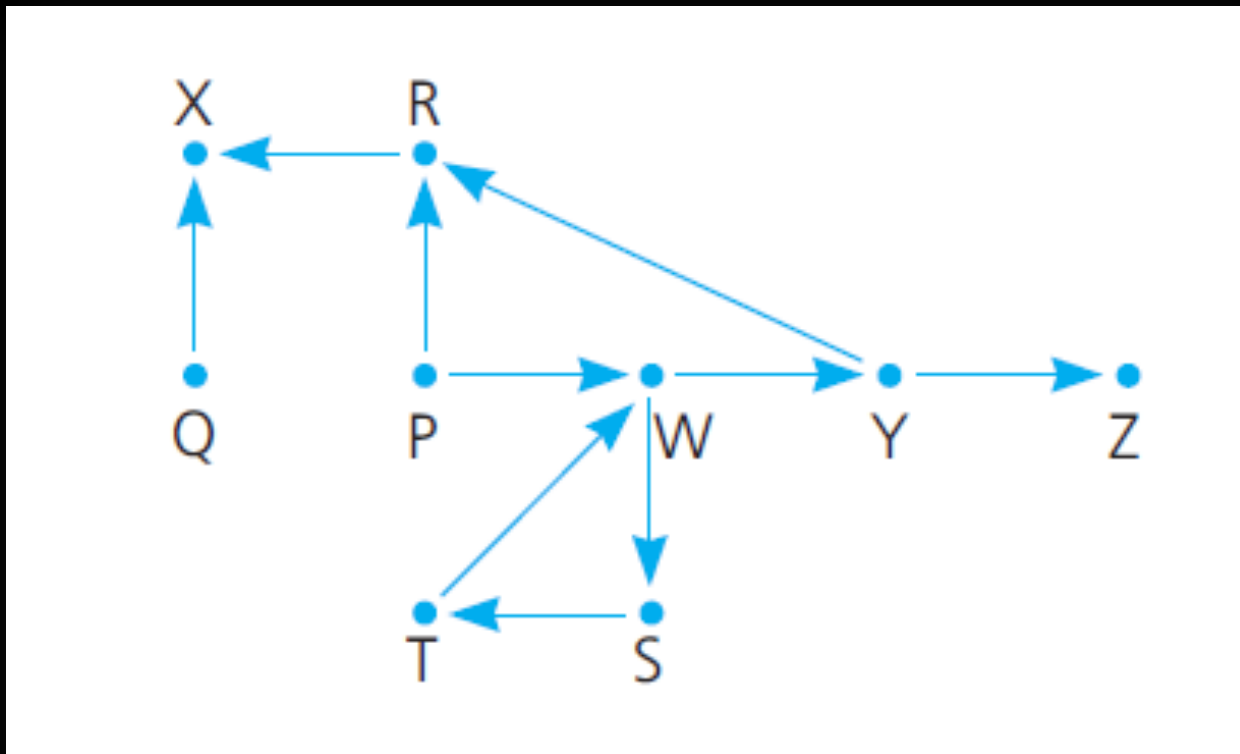


Backtracking

Avoid dead end by backtracking

Avoid traveling in circles by marking visited cities

Origin = P , **Destination = Z**

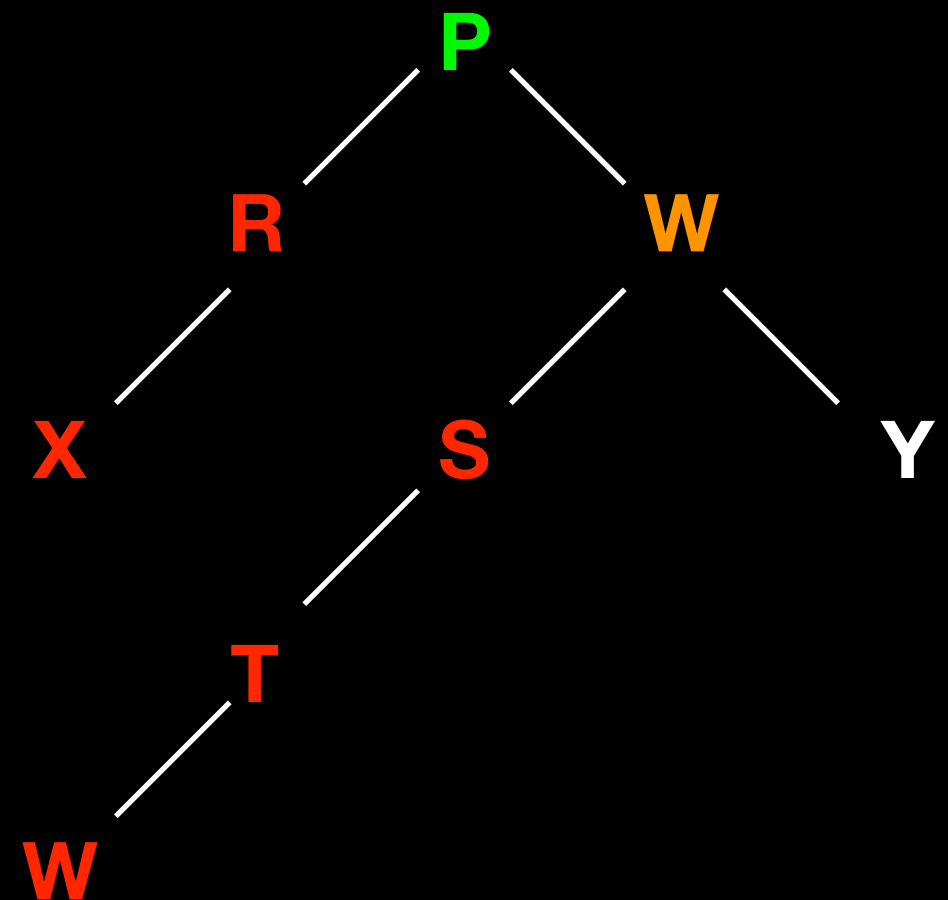
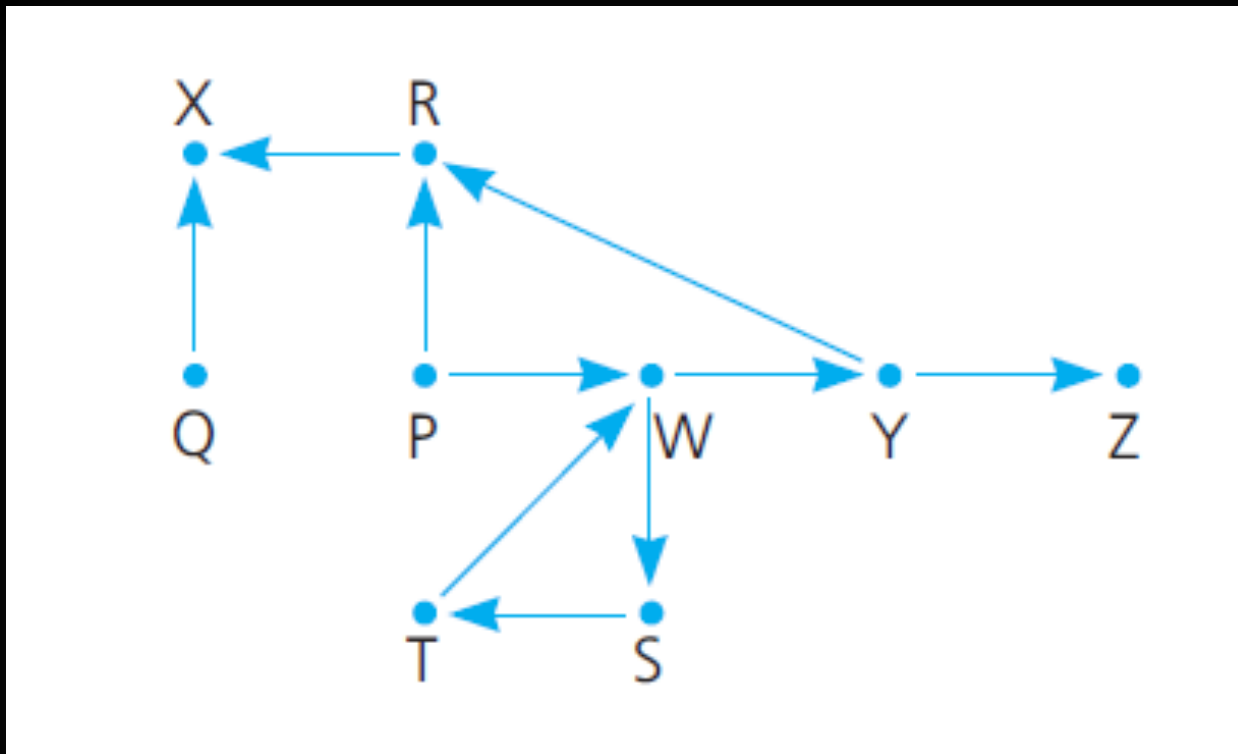


Backtracking

Avoid dead end by backtracking

Avoid traveling in circles by marking visited cities

Origin = P , **Destination = Z**

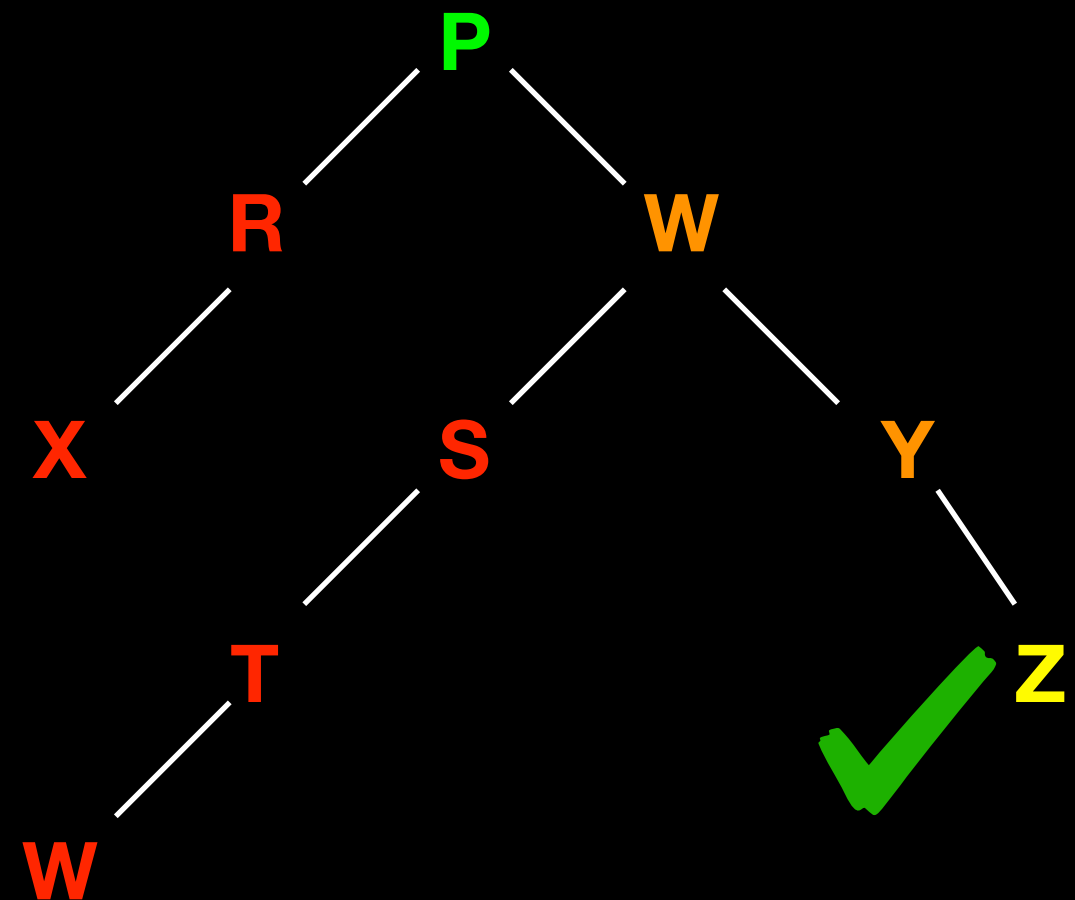
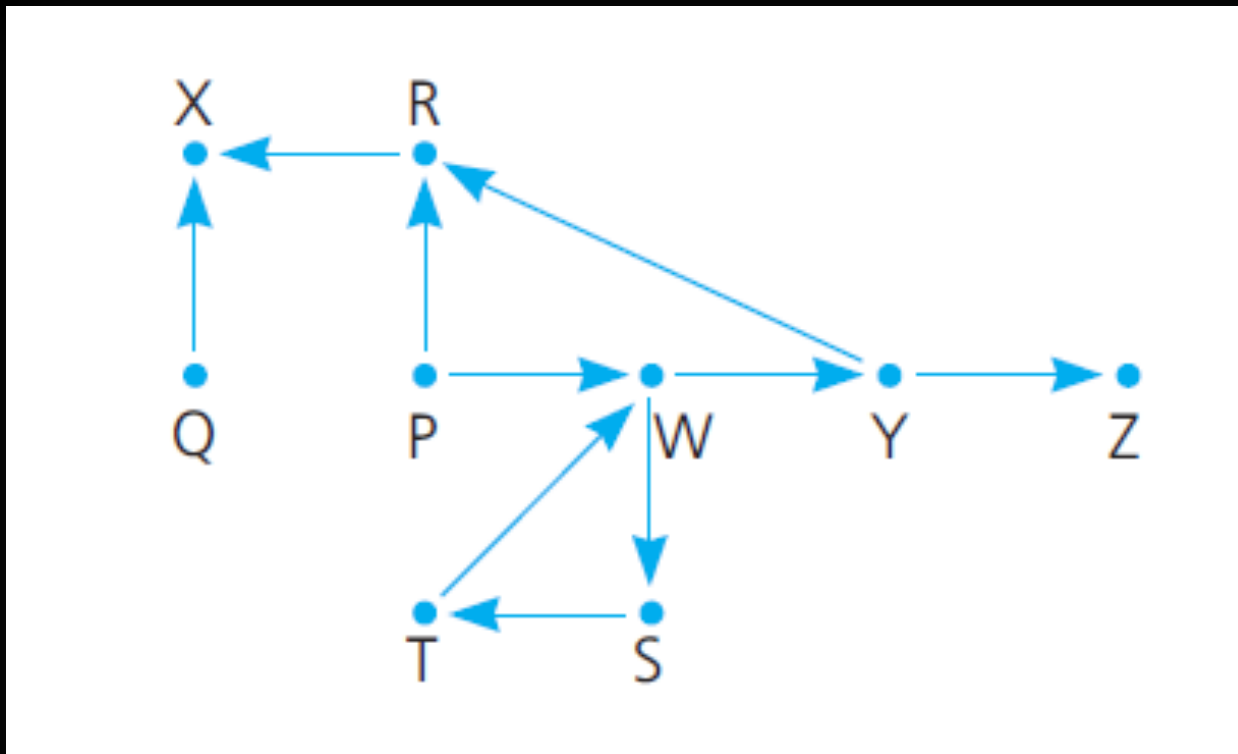


Backtracking

Avoid dead end by backtracking

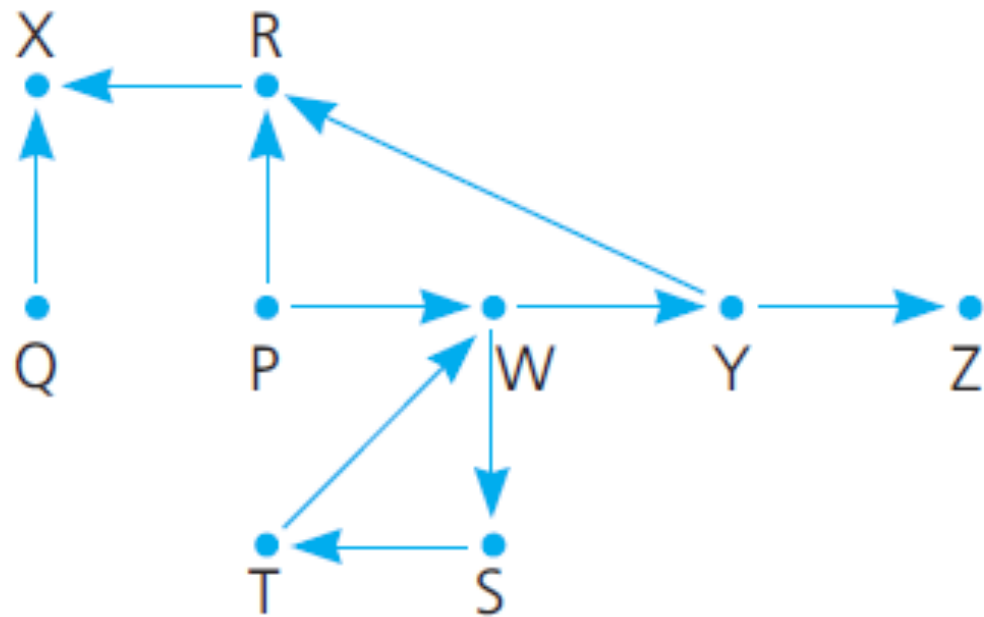
Avoid traveling in circles by marking visited cities

Origin = P , **Destination = Z**



Backtracking

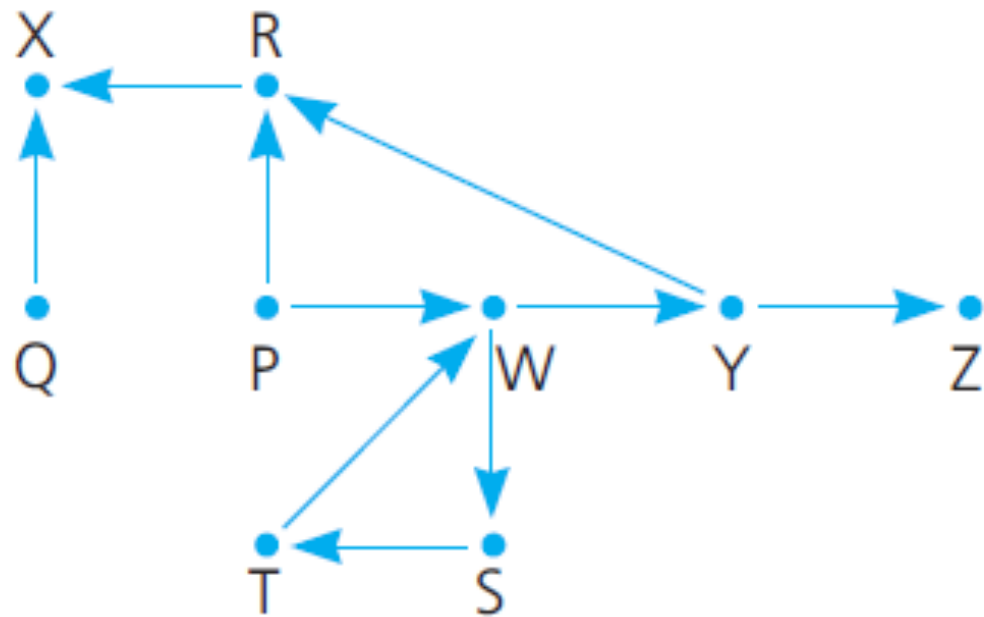
Origin = P , Destination = Z



P

Backtracking

Origin = P , Destination = Z

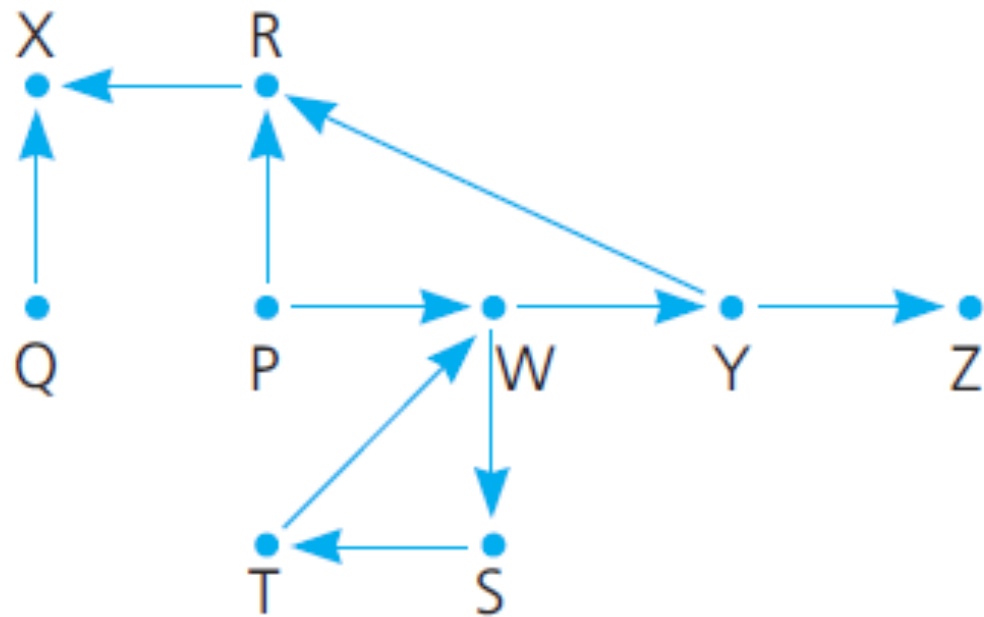


R

P

Backtracking

Origin = P , Destination = Z



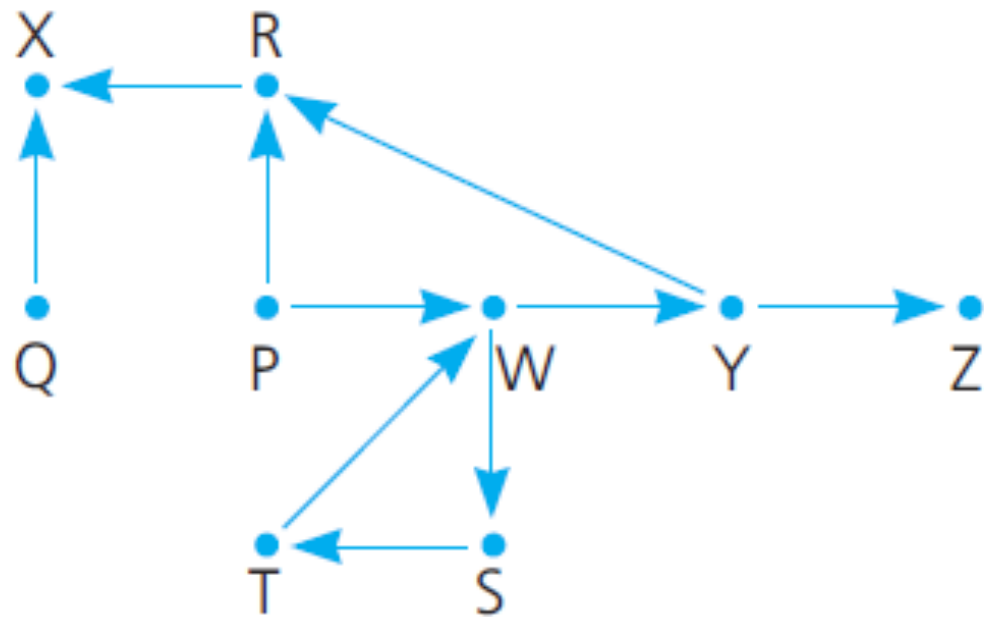
X

R

P

Backtracking

Origin = P , Destination = Z

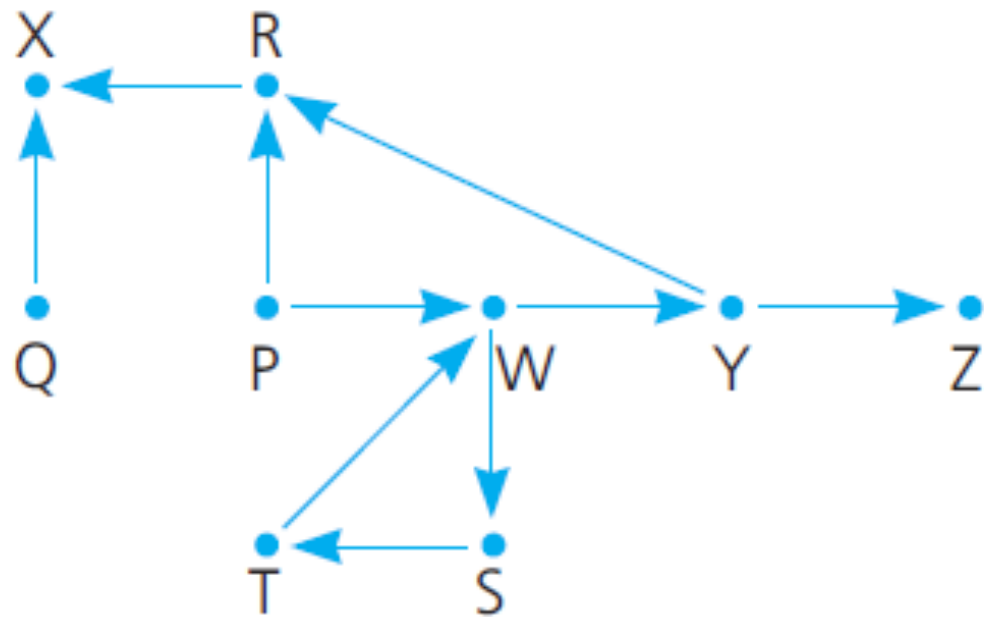


R

P

Backtracking

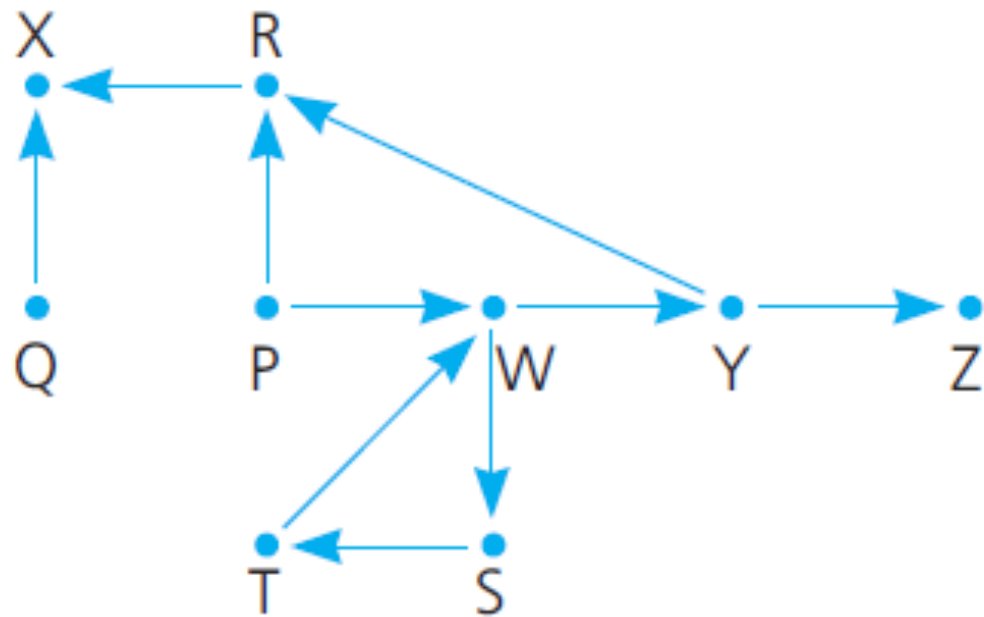
Origin = P , Destination = Z



P

Backtracking

Origin = P , Destination = Z

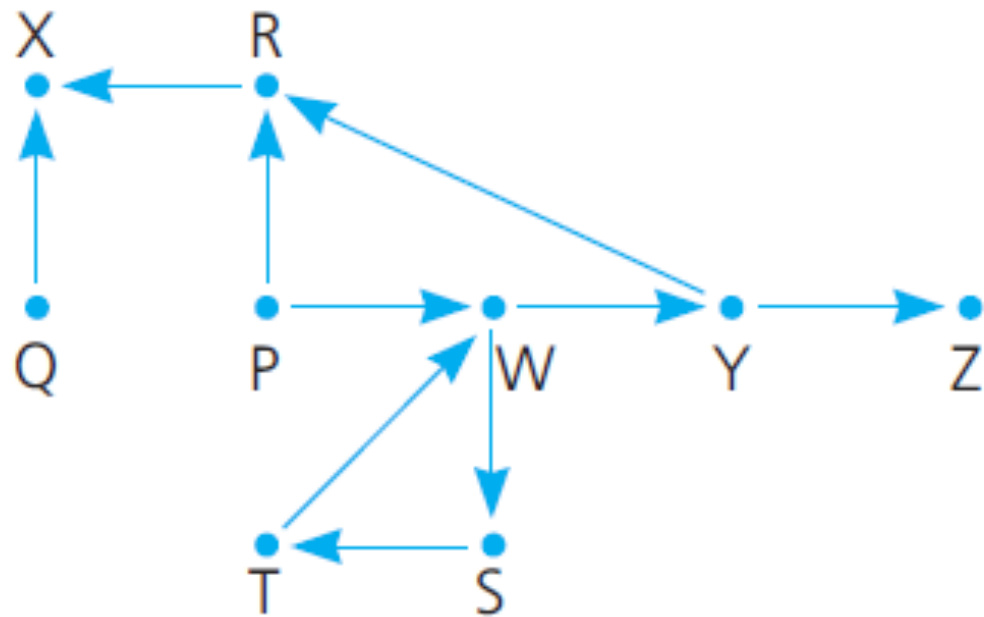


W

P

Backtracking

Origin = P , Destination = Z



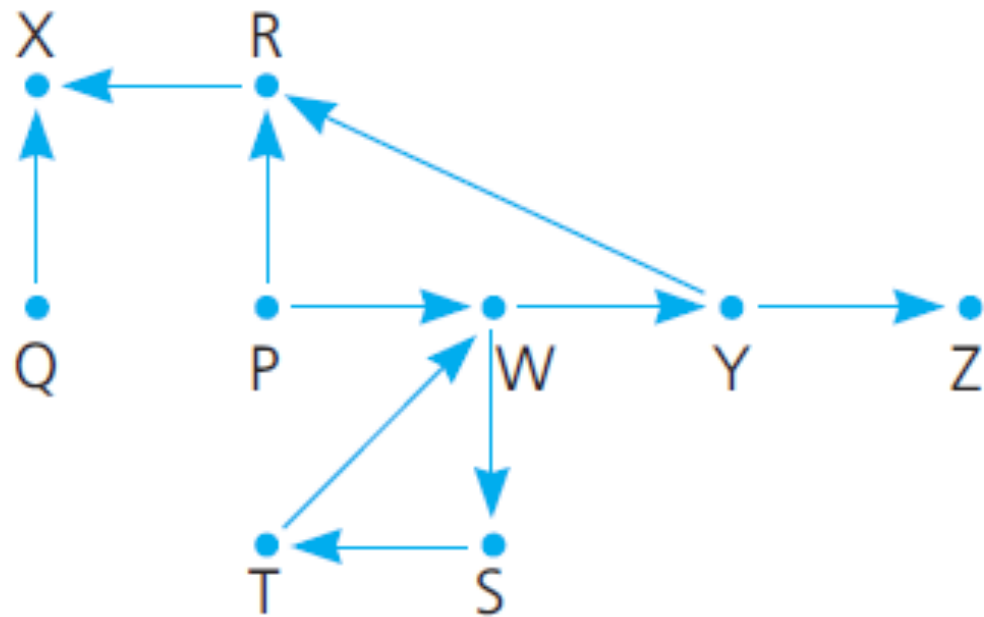
S

W

P

Backtracking

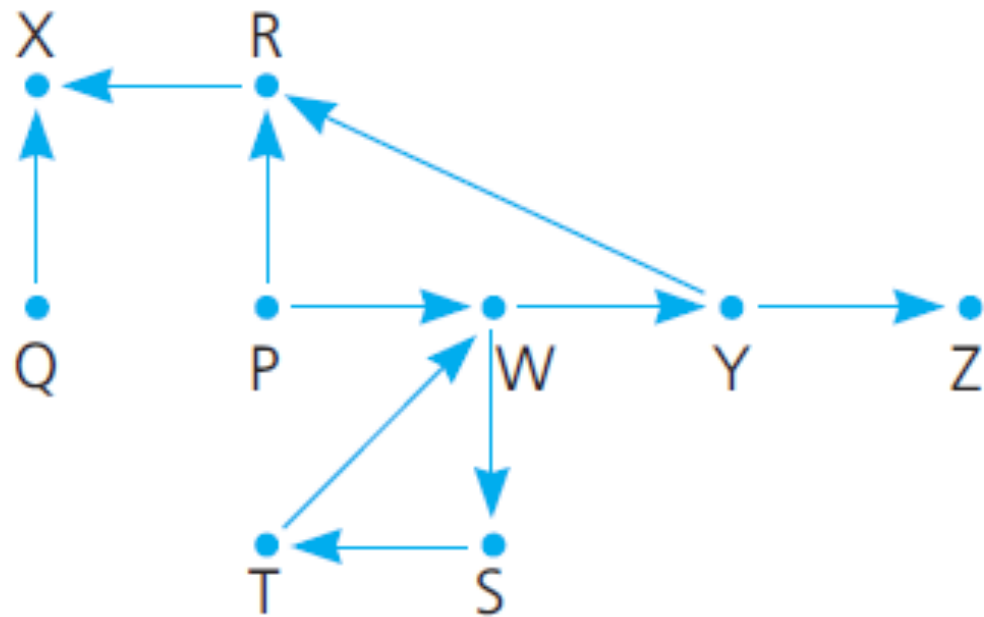
Origin = P , Destination = Z



T
S
W
P

Backtracking

Origin = P , Destination = Z



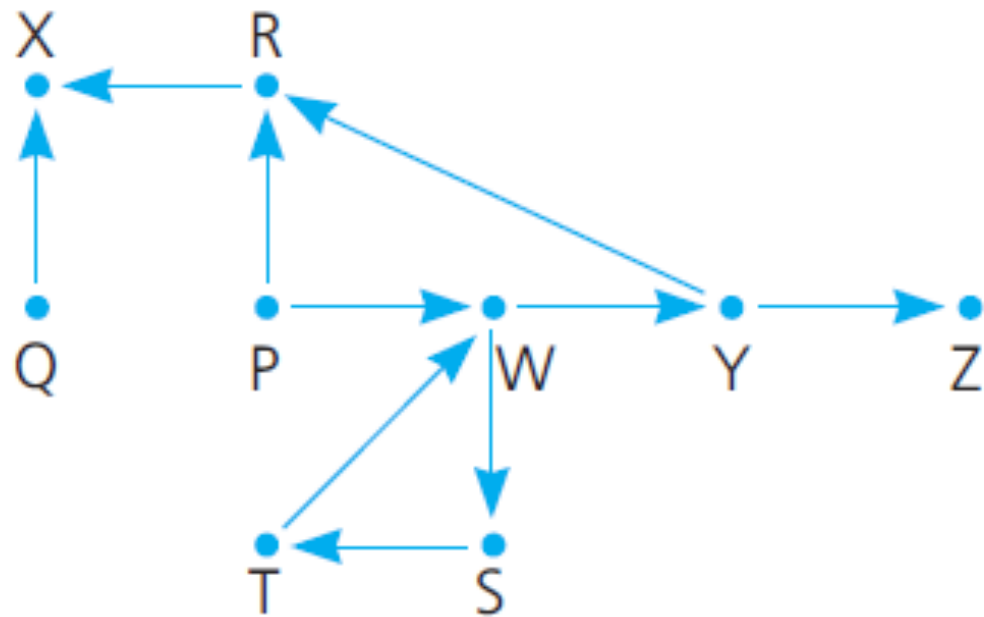
S

W

P

Backtracking

Origin = P , Destination = Z

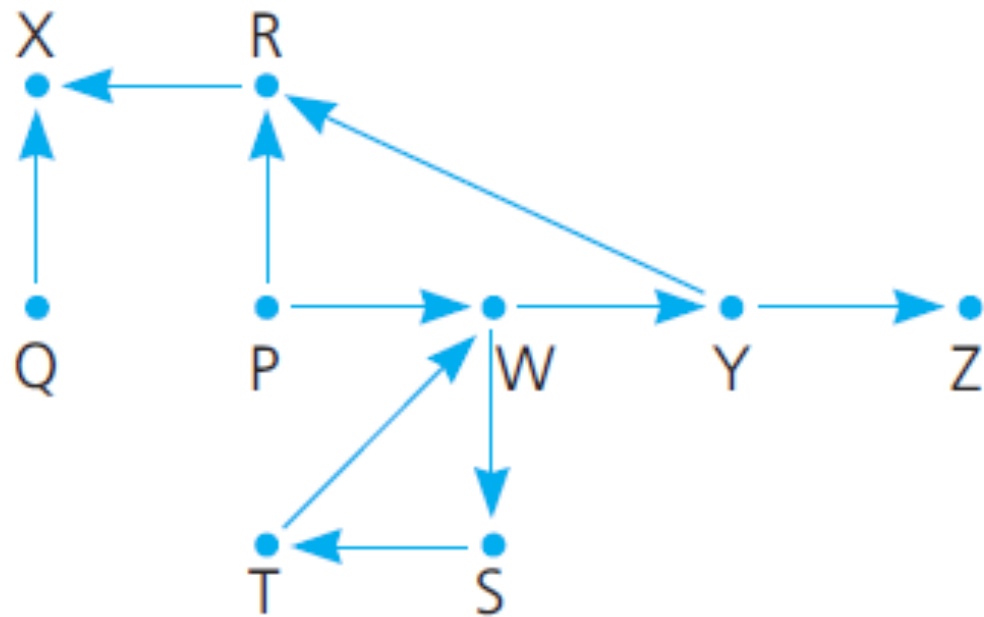


W

P

Backtracking

Origin = P , Destination = Z



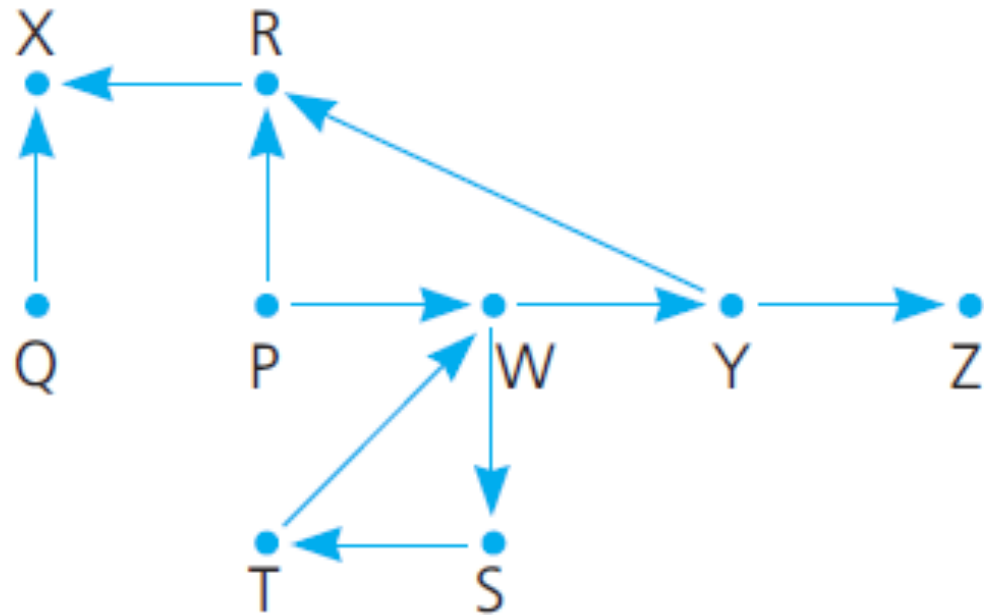
Y

W

P

Backtracking

Origin = P , Destination = Z



Z

Y

W

P



Backtracking

```
while(not found flights from origin to destination)
{
    if no flight exists from city on top of stack to
    unvisited destination
        pop the stack //BACKTRACK
    else
    {
        select an unvisited city C accessible from city
        currently at top of stack
        push C on stack
        mark C as visited
    }
}
```

More Applications

Balancing anything!

-html tags (e.g `<p>` matches `</p>`)

Reverse characters in a word or words in a sentence

Undo mechanism for editors or backups

Traversals (graphs / trees)

...

Interface Note

You can use inheritance from an abstract data type class to enforce a public interface

Complete separation of interface from implementation

You know how to do it, we will no longer do it

Stack ADT

```
#ifndef STACK_H_
#define STACK_H_

template<class ItemType>
class Stack
{
public:
    Stack();
    void push(const ItemType& newEntry); // adds an element to top of stack
    void pop(); // removes element from top of stack
    ItemType top() const; // returns a copy of element at top of stack
    int size() const; // returns the number of elements in the stack
    bool isEmpty() const; // returns true if no elements on stack false otherwise

private:
    //implementation details here

}; //end Stack

#include "Stack.cpp"
#endif // STACK_H_`
```