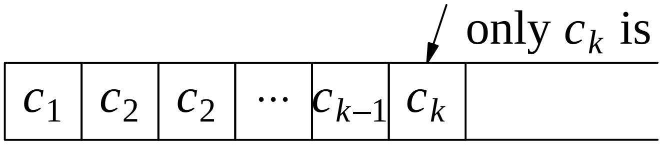
PUSH-DOWN MEMORY AND PUSH-DOWN AUTOMATA

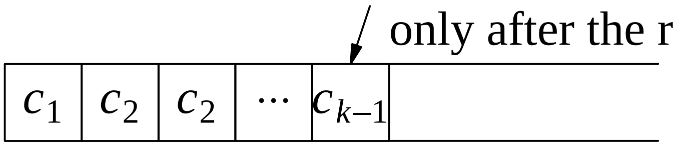
Push-down (stack) memory:

top (most recent) item in memory;
only c_k is accessible at this point



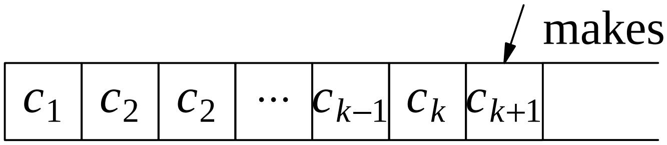
stack = $c_1 c_2 \cdots c_{k-1} c_k$.

c_{k-1} is accessible
only after the removal of c_k



stack = $c_1 c_2 \cdots c_{k-1}$

addition of c_{k+1}
makes c_k inaccessible



stack = $c_1 c_2 \cdots c_{k-1} c_k c_{k+1}$

Push-down automata = FSA + push-down memory.

- The stack-alphabet may be different from input alphabet.

Each transition does four things:

(1) Reads an input-symbol (and moves the read-head right).

(2) Looks at the top stack-symbol.

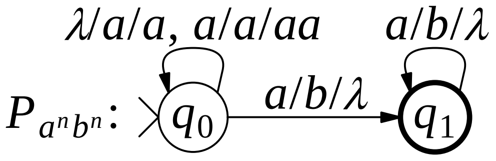
(3) Replaces the top stack-symbol by zero or more symbols.

(4) Changes state (perhaps back to the same state).

EXAMPLE OF A PDA

Example. A PDA for $L = \{a^n b^n : n \geq 1\}$.

- Save the *a*'s on the stack, then match them off with the *b*'s.



Deterministic PDA

- Transition-label: (stack-symbol, input-symbol, replacement-string).
- Initially stack is empty; in this case, stack-symbol = λ.

Processing of input $x = aaabbb$**:**

Stack (rightmost symbol on top)	State	Remaining input string	Transition used
λ	q_0	$aaabbb$	$\lambda/a/a$
a	q_0	$aabbb$	$a/a/aa$
aa	q_0	$abbb$	$a/a/aa$
(end of saving *a*'s) aaa	q_0	bbb	$a/b/\lambda$
aa	q_1	bb	$a/b/\lambda$
a	q_1	b	$a/b/\lambda$
(end of matching *b*'s) λ	q_1	λ	

Language Defined by PDA: $\{x : x \text{ is accepted}\}$.

x is accepted: The PDA reaches a final-state on reading all of *x* *and* the stack is empty at that point.

x is rejected: *x* cannot be read completely *or* after reading *x*, PDA does not reach a final-state or the stack is not empty.

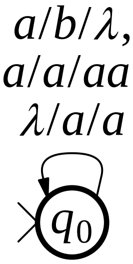
PDA FOR BALANCED PARENTHETICAL STRINGS

We write *a* for ’(’ and *b* for ’)’for readability.

$$L_{\lambda+bal\text{-}par} = \{\lambda, ab, abab, aabb,\ ababab, abaabb, aabbab, aababb, aaabbb, \cdots\}$$

- Each of these strings, except for λ, contains an initial string of the form $a^n b^n$ ($n > 0$) or has such a string following an initial part a^m.
- The removal of that $a^n b^n$ makes the smaller string again balanced.

Idea: Merge q_0 and q_1 in $P_{a^n b^n}$ into a single state to allow the *saving* of *a*’s and *matching* off with *b*’s can occur repeatedly.



$P_{\lambda+bal\text{-}par}$:

Deterministic PDA

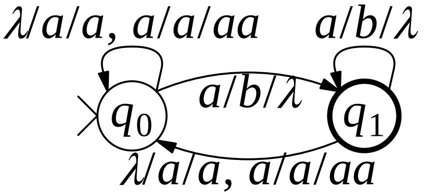
Processing of $x = a\cdot\underline{ab}\cdot\underline{ab}\ \ b$**:**

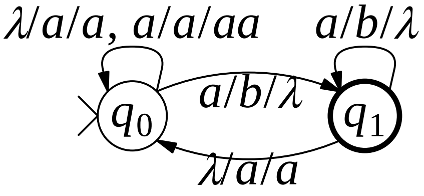
Stack memory	State	Remaining input string	Transition used
λ	q_0	*aababb*	$\lambda/a/a$
a	q_0	*ababb*	$a/a/aa$
aa	q_0	*babb*	$a/b/\lambda$
a	q_0	*abb*	$a/a/aa$
aa	q_0	*bb*	$a/b/\lambda$
a	q_0	*b*	$a/b/\lambda$
λ	q_0	λ	

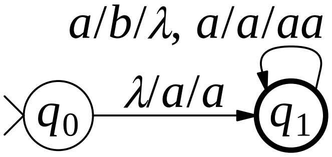
EXERCISE.

1. Construct a PDA to accept $L_{bal\text{-}par}$; $\lambda \notin L_{bal\text{-}par}$.

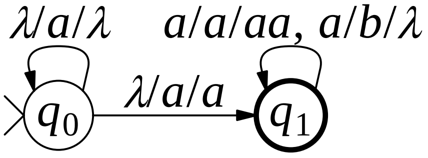
2. For each PDA below, describe the language accepted by it. Which of the PDAs are deterministic?

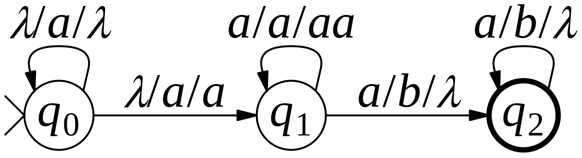


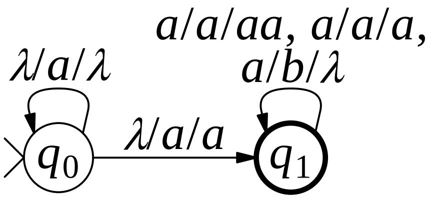


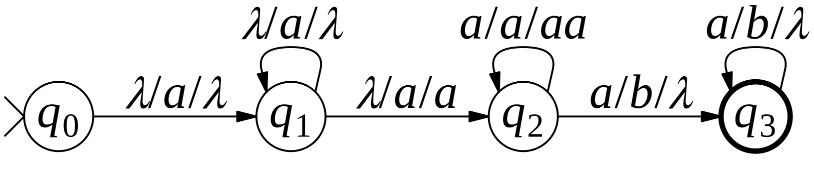


3. Consider the language $L_{m \geq n} = \{a^m b^n\colon m \geq n \geq 1\}$ and the PDAs below. For each PDA, find a smallest string $x \in L_{m \geq n}$ not accepted or $x \notin L_{m \geq n}$ but accepted. For each PDA, eliminate any unnecessary non-determinism if possible (without changing the language of the PDA).









4. Show a PDA which accepts all non-empty strings over the alphabet {0, 1, ×, =} which represent correct multiplications of the form $2^m 2^n = 2^{m+n}$. For $m = 2$ and $n = 3$, the input string is 100×1000=100000. (Hint: the symbols ’×’ and ’=’ in the input string should drive the PDA to different states.)

5. Show that the language described in Problem 4 is not regular.

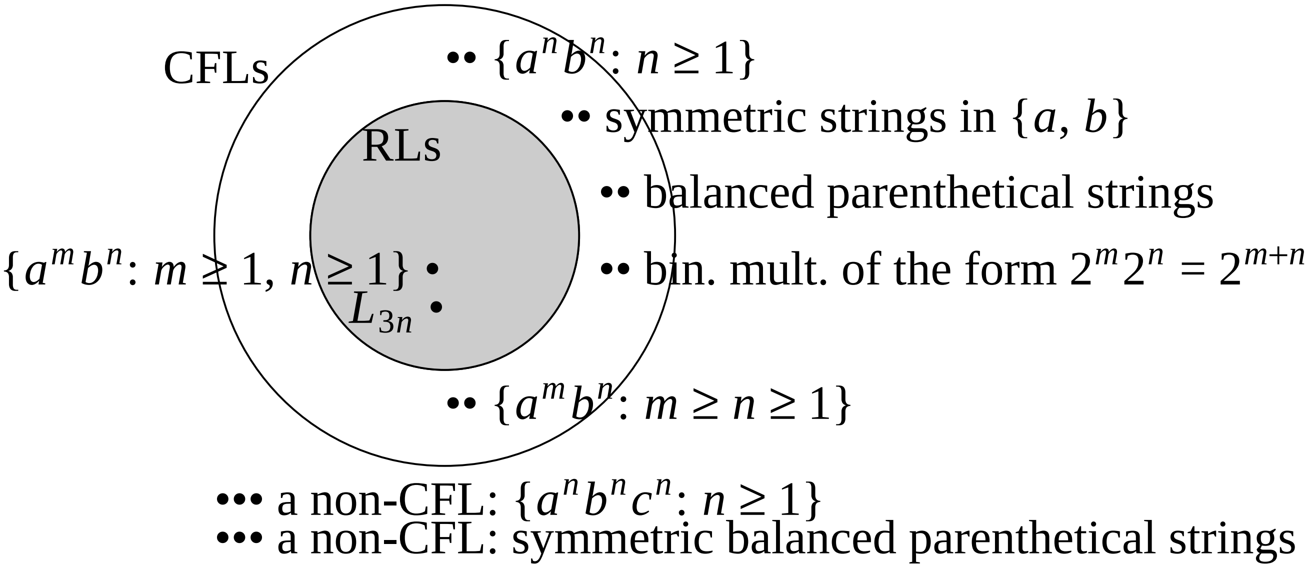
6. Give the state-diagram for a deterministic PDA which accepts the language $L_{m=2n} = \{a^m b^n\colon m = 2n,\ n \geq 1\}$. Do the same for the language $L_{m=2n+1}$, defined in a similar way.

CONTEXT-FREE LANGUAGES AND PDAs

CFL: A language is context-free if it accepted by a PDA.

Language point of view:

- CFLs describe more general patterns than those described by regular languages. (Each regular language is also a CFL.)


CFLs
RLs
•• $\{a^n b^n : n \geq 1\}$
•• symmetric strings in $\{a, b\}$
•• balanced parenthetical strings
•• bin. mult. of the form $2^m 2^n = 2^{m+n}$
$\{a^m b^n : m \geq 1, n \geq 1\}$ •
L_{3n} •
•• $\{a^m b^n : m \geq n \geq 1\}$
••• a non-CFL: $\{a^n b^n c^n : n \geq 1\}$
••• a non-CFL: symmetric balanced parenthetical strings

Machine point of view:

- PDAs characterize computations that can be performed by a limited use of *external* memory, called the *stack* memory, where the amount of memory used is no more than a constant times the size of the input.

A PDA uses at most a linear
(in the size of the input) amount of stack memory.

PDA FOR SYMMETRIC STRINGS

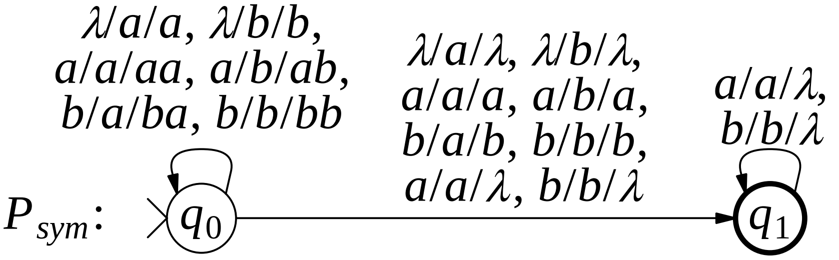
L_{sym} = all non-empty symmetric strings over $\{a, b\}$.
= $\{a, b, aa, bb, aaa, aba, bab, bbb, \cdots\}$

Idea:

(1) Non-deterministically decide a mid-point of the string (and changes state from q_0 to q_1).

(2) Non-deterministically decide whether the string is of even length or odd length. In the second case, ignore the unique middle symbol (not matched with the next symbol).

(3) Save the first half of the string on the stack, and then match it off with the second half.



3 transitions at q_0 for (stack-symbol, input-symbol) = (a, a).

Processing of $x = aabbaa$ (*deterministic* accepting behavior).

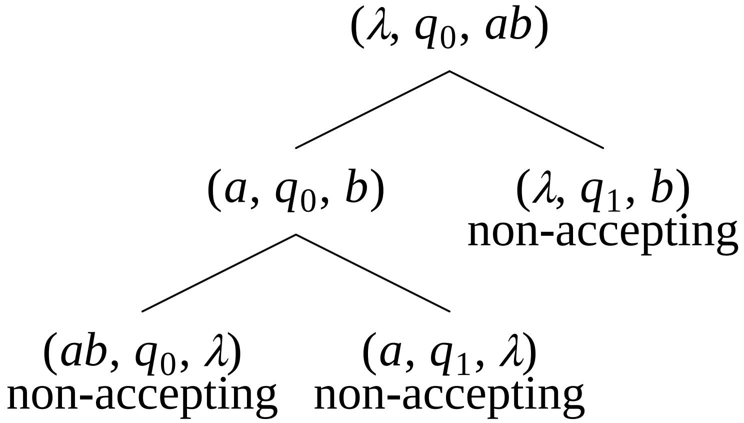
Stack memory	State	Remaining	Transition
	tate	input string	used
λ	q_0	$aabbaa$	$\lambda/a/a$
a	q_0	$abbaa$	$a/a/aa$
aa	q_0	$bbaa$	$a/b/ab$
aab	q_0	baa	$b/b/\lambda$
aa	q_1	aa	$a/a/\lambda$
a	q_1	a	$a/a/\lambda$
λ	q_1	λ	

Any other processing lead to rejection of *aabbaa*.

ALL MOVE-SEQUENCES IN P_{sym} FOR $x = ab \notin L_{sym}$

Configuration: (stack, state, remaining-input).

- For input x and start-state = q_0, initial configuration is (λ, q_0, x).



$x = ab$ is not accepted by P_{sym}.

EXERCISE.

1. Give the numbers of symmetric strings and non-symmetric strings for length $m = 2k$ and length $m = 2k + 1$.
2. Give the state-diagram of a deterministic PDA for accepting the language $L_{\#a=\#b} = \{x \in (a + b)^+ : \#(a, x) = \#(b, x)\}$.
3. Give a PDA for $L_{a^{n+3}b^n}$.
4. Give the state diagram of deterministic-behavior PDA for $L_{non-sym}$ = $\{ab, ba, aab, abb, baa, bba, \cdots\}$ with deterministic-behavior. Show the tree of all move sequences of the PDA for $x = ab$.
5. Show a PDA for the language $L_{m \neq n} = \{a^m b^n : m \neq n, m, n \geq 1\}$ with deterministic behavior.

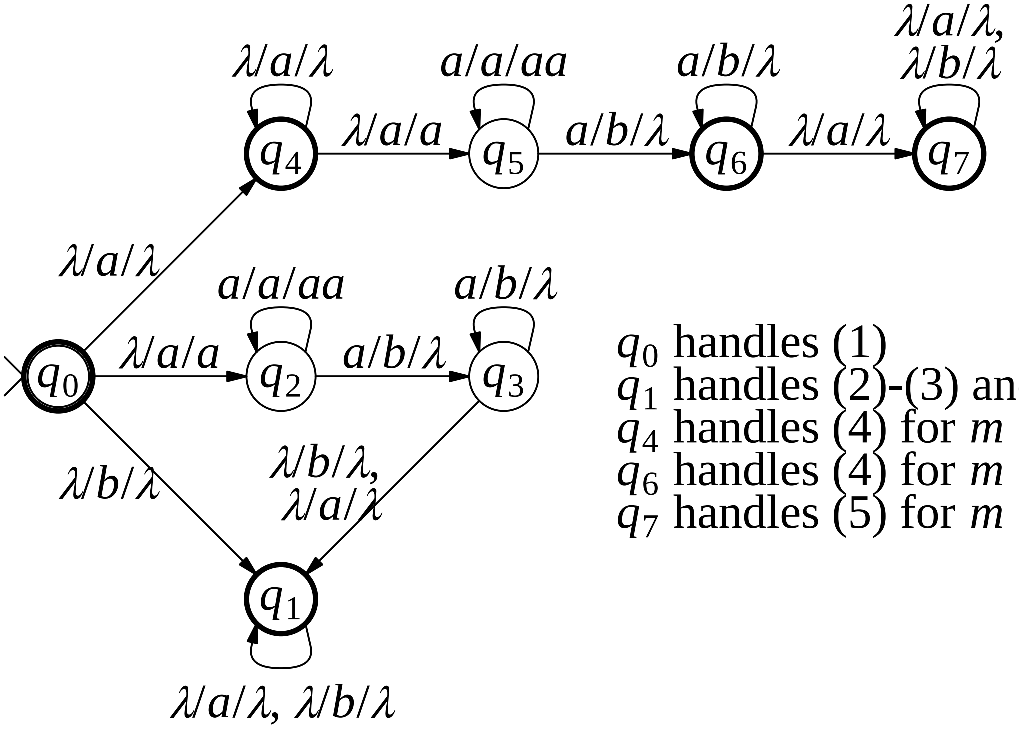
   Show the configurations in all possible move sequences for your PDA for the input $x = a^2 b^2$ to show that it is not accepted.

COMPLEMENT OF $\{a^n b^n\text{: } n \geq 1\}$

Five kinds of strings in the complement:

(1) $x = \lambda$.
(2) x begins with b, i.e., $x = by$.
(3) $x = a^n b^{n+1} y$ ($n > 0$).

(4) $x = a^n b^m$ and $m < n$ (> 0).
(5) $x = a^n b^m ay$ and $m \leq n$ (> 0).



q_0 handles (1)
q_1 handles (2)-(3) and (5) for $m = n$
q_4 handles (4) for $m = 0$
q_6 handles (4) for $m > 0$
q_7 handles (5) for $m < n$

Non-deterministic PDA, with a deterministic accepting behavior.

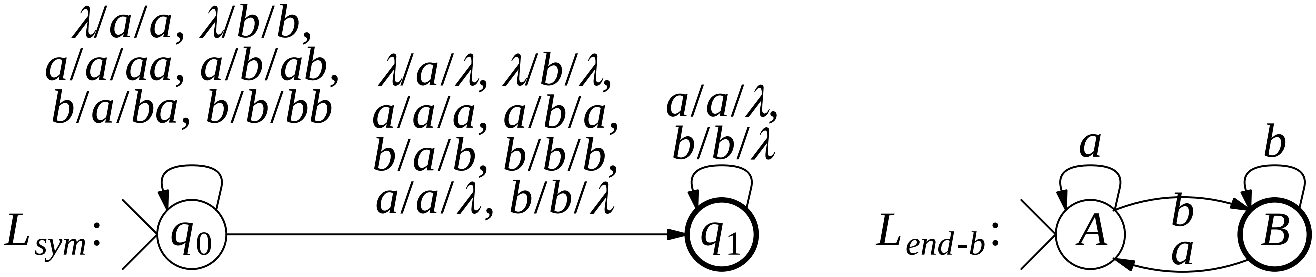
EXERCISE.

1. Construct a PDA to accept the complement of $L_{bal\text{-}par}$; the complement includes λ.

INTERSECTION OF CFL AND RL IS CFL

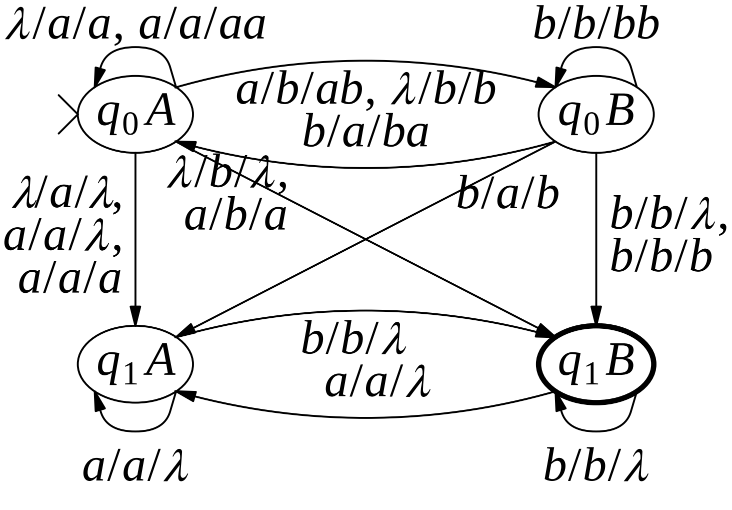
AND-construction of a PDA and an FSA:

- The states of the new PDA have the form $q_i q'_j$, where q_i = a state of the PDA and q'_j = a state of the FSA. The parts q_i and q'_j change according to the PDA and the FSA, respectively.
- The stack operations remain as in the old PDA.



PDA for $L_{sym} \cap L_{end\text{-}b}$:

- Non-deterministic transitions but deterministic behavior.
- The transition $\lambda/a/\lambda$ from q_0A to q_1A never takes part in accepting a string (and can be removed). Are there other such transitions?



Question:

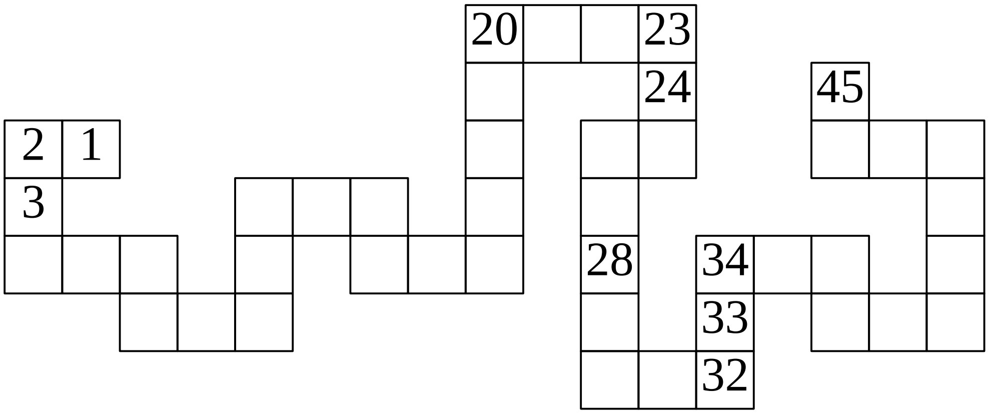
•? Show the tree of all configurations in processing *aba*.

EXERCISE

1. Argue that the union of a CFL and an RL is a CFL.
2. Assume that $L_{a^nb^nc^n} = \{a^nb^nc^n\colon n \geq 1\}$ is not CFL, i.e., it is not accepted by any PDA. Then, show that $\{x \in (a+b+c)^*\colon \#(x, a) = \#(x, b) = \#(x, c)\}$ is not CFL.
3. Give a PDA for the language $L_{\#a=\#b} = \{x \in (a+b)^*\colon \#(x, a) = \#(x, b)\}$.
4. If L is a CFL and L' is regular, then argue that LL' and $L'L$ are CFL
5. Since $L_{a^nb^nc^n} = L_{a^nb^nc^*} \cap L_{a^*b^nc^n}$ and both $L_{a^nb^nc^*}$ and $L_{a^*b^nc^n}$ are CFL, it follows that intersection of CFLs need not be a CFL. Thus, complement of a CFL need not be a CFL either.

FINDING NORTHMOST POSITION IN A LINEAR MAZE

Linear Maze: Each square has ≤ 2 neighbors; it is a strip laid in a straight line or a curved non-crossing fashion.



Some observations:

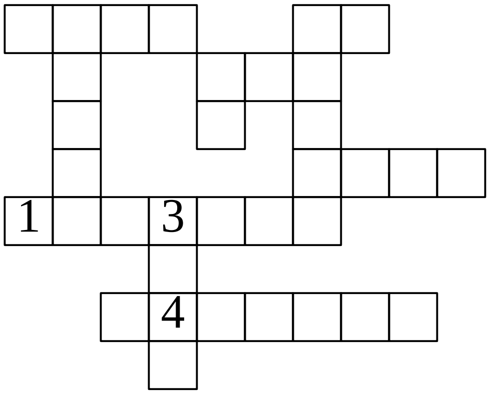
- Starting at 1, when the robot reaches 20 (or 23) the stack must tell us that the robot is one step further north of position 1.
- At 28 (also at 34), the stack should tell us that the maximum north reached so far was one step higher than the start position, and that currently the robot is 4 steps south from that max-north position.
- At 45, the stack must tell us we are still south of the max-north position and it must allow us to get back to position 23, say.

EXERCISE.

1. Design a PDA-control of a robot so that for any initital placement of it in a linear maze, it will stop at a northmost position. Hints: (1) First make the robot go to a dead-end. (2) Use the stack to keep track of north-south motion as it now moves to the other dead-end. Show the stack for each position of the robot till it stops.

TRAVERSING AN ACYCLIC MAZE ONCE

1. Design a PDA to control the motion of a robot so that if it is put in any acyclic maze as shown below, then it will start traversing the maze using "keep to the right" strategy and come to a stop when it arrives at its initial position for the first time; in the process it may have traversed the whole maze or only a part of the maze. (Hint: Use the stack to keep track of information related to the path taken so that when you return to the initial position, the stack is empty.) Show the stack during the operation of robot starting at 3 till it gets back to it for the first time.



2. Is it possible to make the robot stop at a northmost position in any acyclic maze?