EMPIRICAL COMPARISON OF TRAVELING SALESPERSON APPROXIMATION ALGORITHMS

Thesis Defense for Shayan Daijavad

Committee: Professor Frishberg, Professor Teo, Professor Ventura

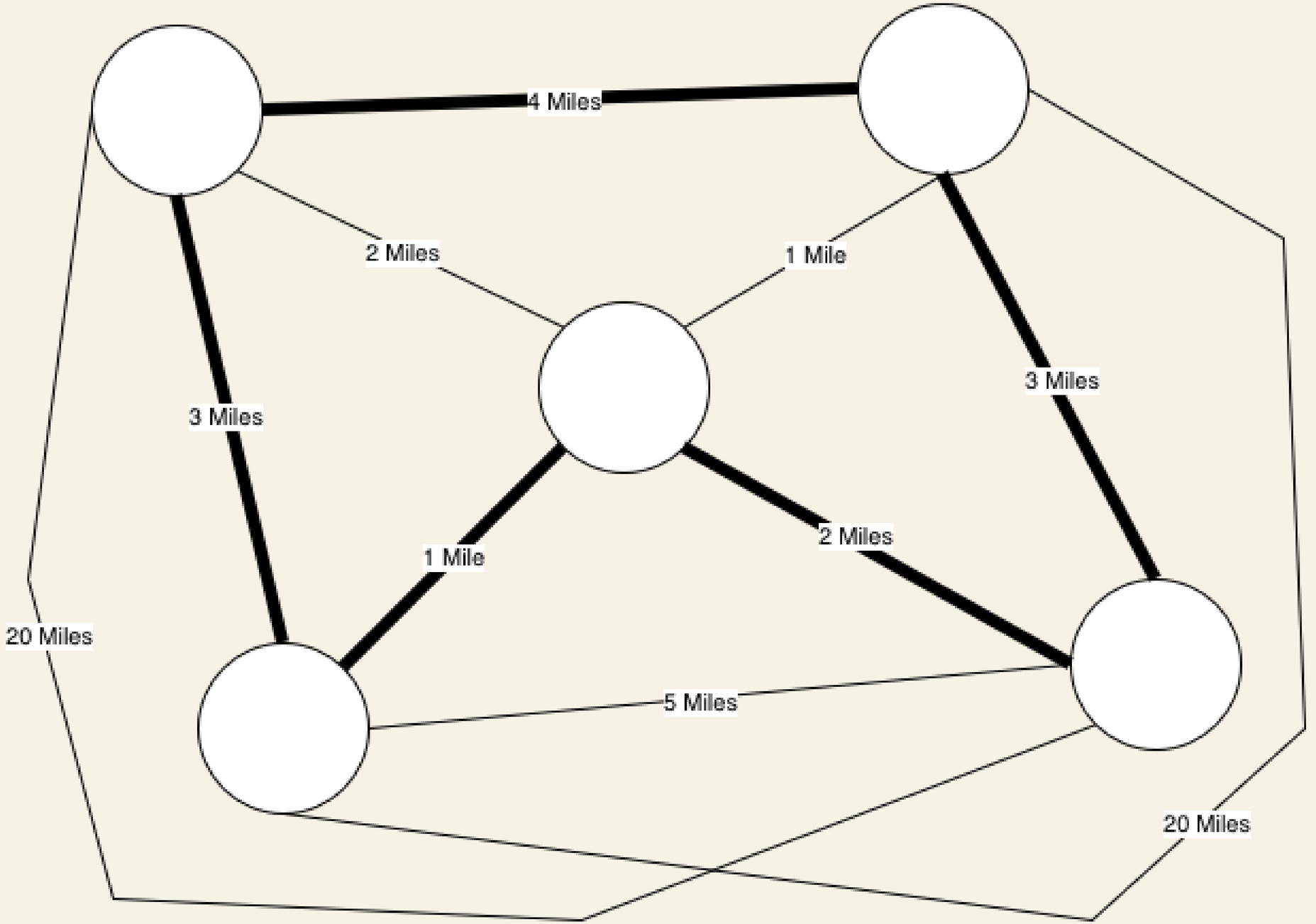
June 9th 2026

Table of Contents

INTRODUCTION

The Traveling Salesperson Problem

- **Definition:** Given a set of n points with pair-wise distances between them, find the shortest tour that visits all n points exactly once and returns back to the starting point of the tour.

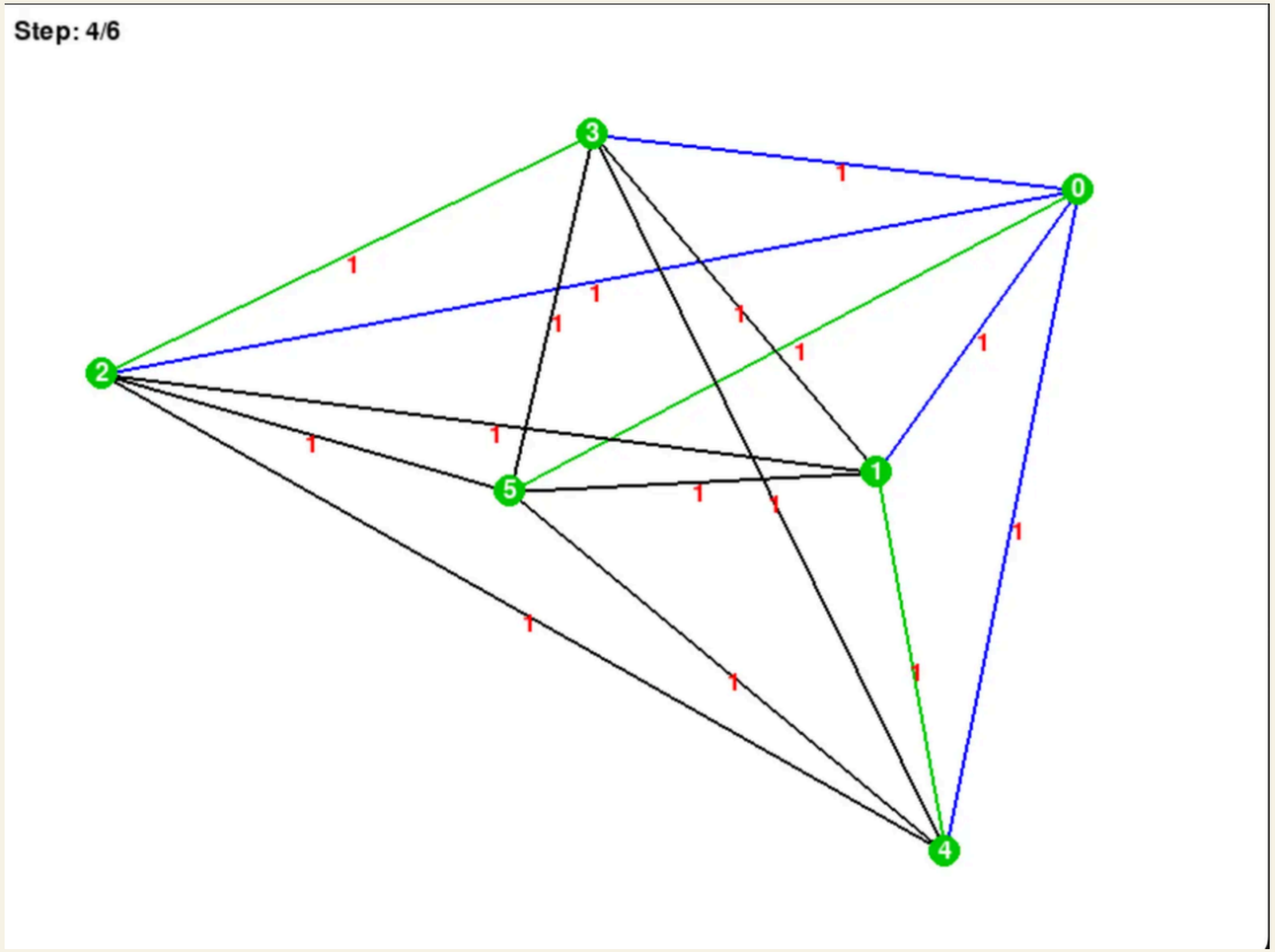

4 Miles
2 Miles
1 Mile
3 Miles
3 Miles
1 Mile
2 Miles
20 Miles
5 Miles
20 Miles

Origins + Applications

- Karl Menger
- Vehicle routing
- Circuit board drilling
- Computer wiring
- Molecular biology

My Interest in the Problem

- CSC 349 with Professor Migler
- CSC 549 with Professor Frishberg
- Mamano et al. paper



Two Algorithms

- Multifragment Heuristic with Nearest Neighbor Chain
  - O(nlogn) runtime
  - O(logn) approximation
- 2-Approximation
  - $O(n^2)$ runtime
  - 2OPT approximation

Research Questions

- RQ1: In practice, what are the tradeoffs in approximation performance, runtime, and implementation simplicity between these algorithms?
- RQ2: In theory, NNC gives a speedup for the MFH. In practice, how much of a speedup is there?

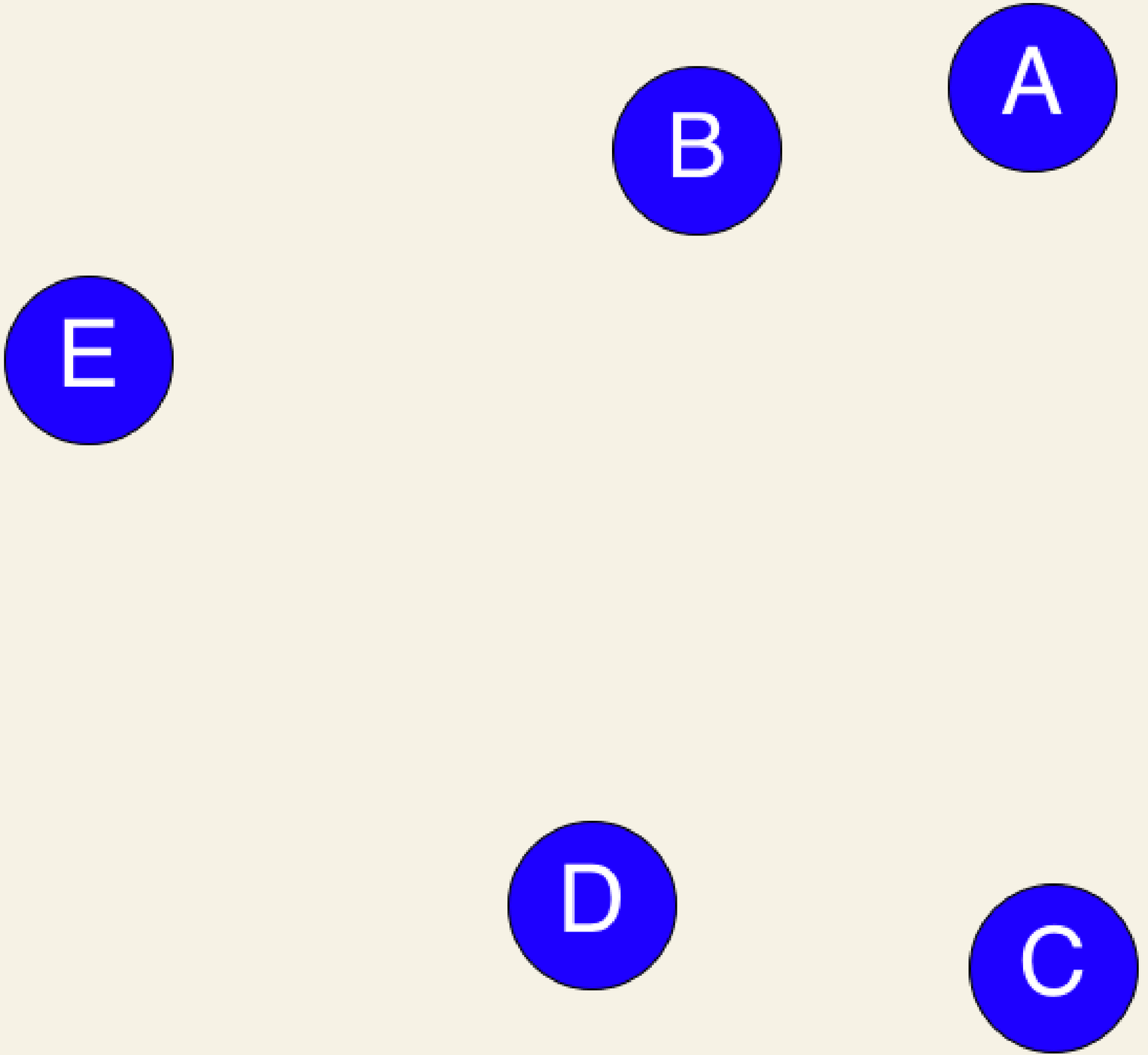
IMPLEMENTATION

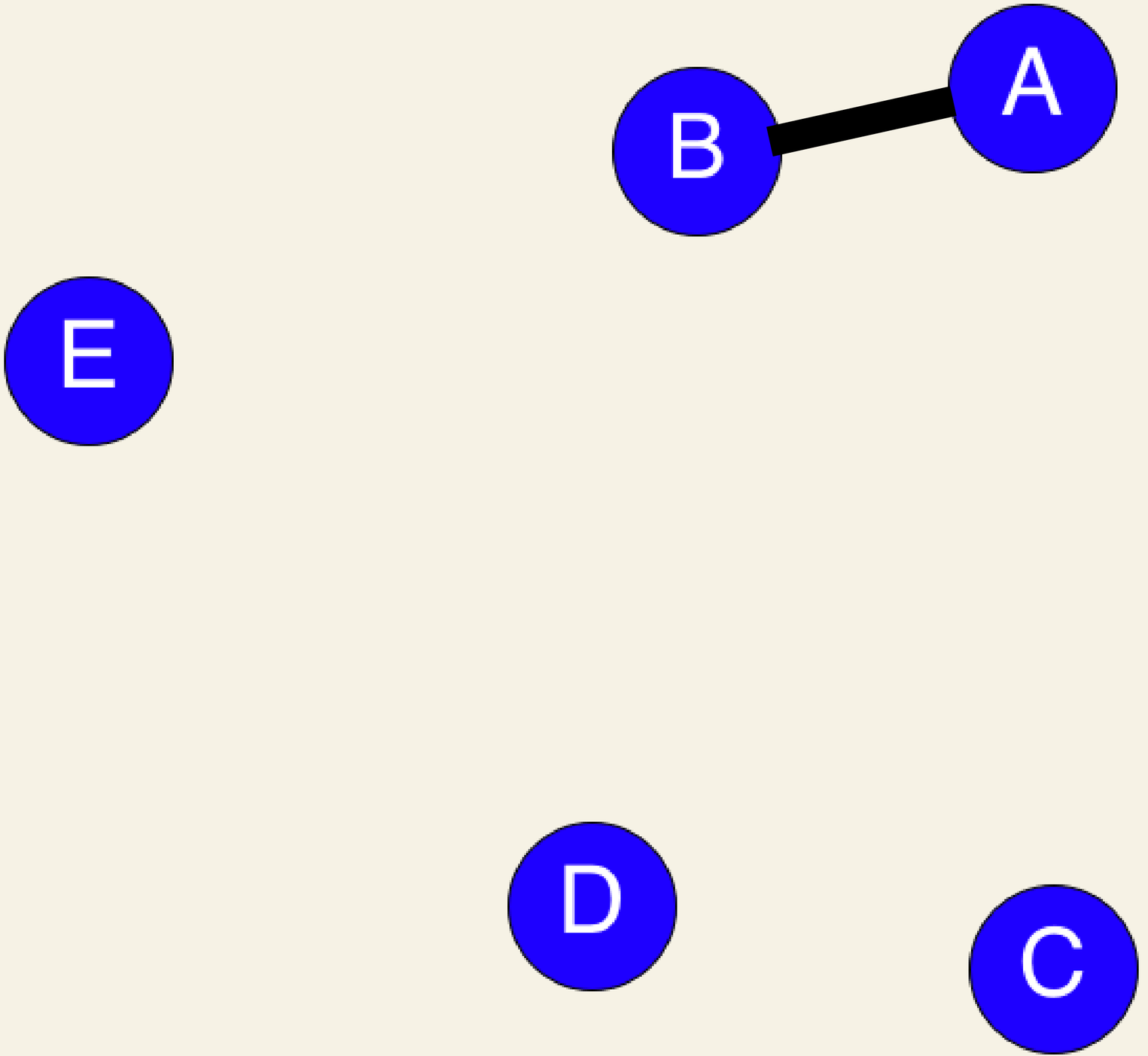
Background Details

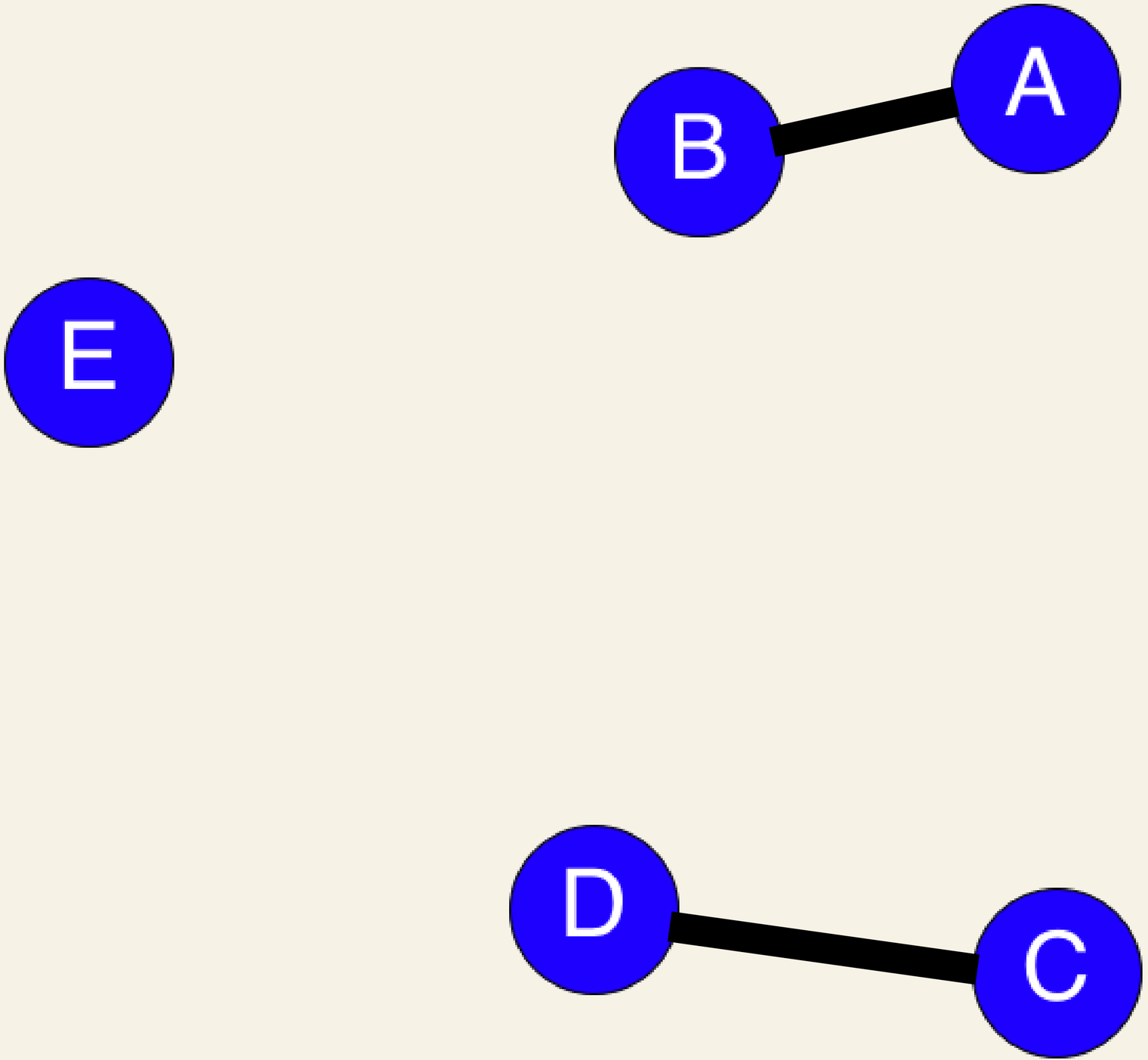
- C++ for speed
- Timeline:
  - Naive MFH Implementation
  - NNC MFH Implementation
  - Christofides
  - Pivot
  - 2-Approximation Implementation

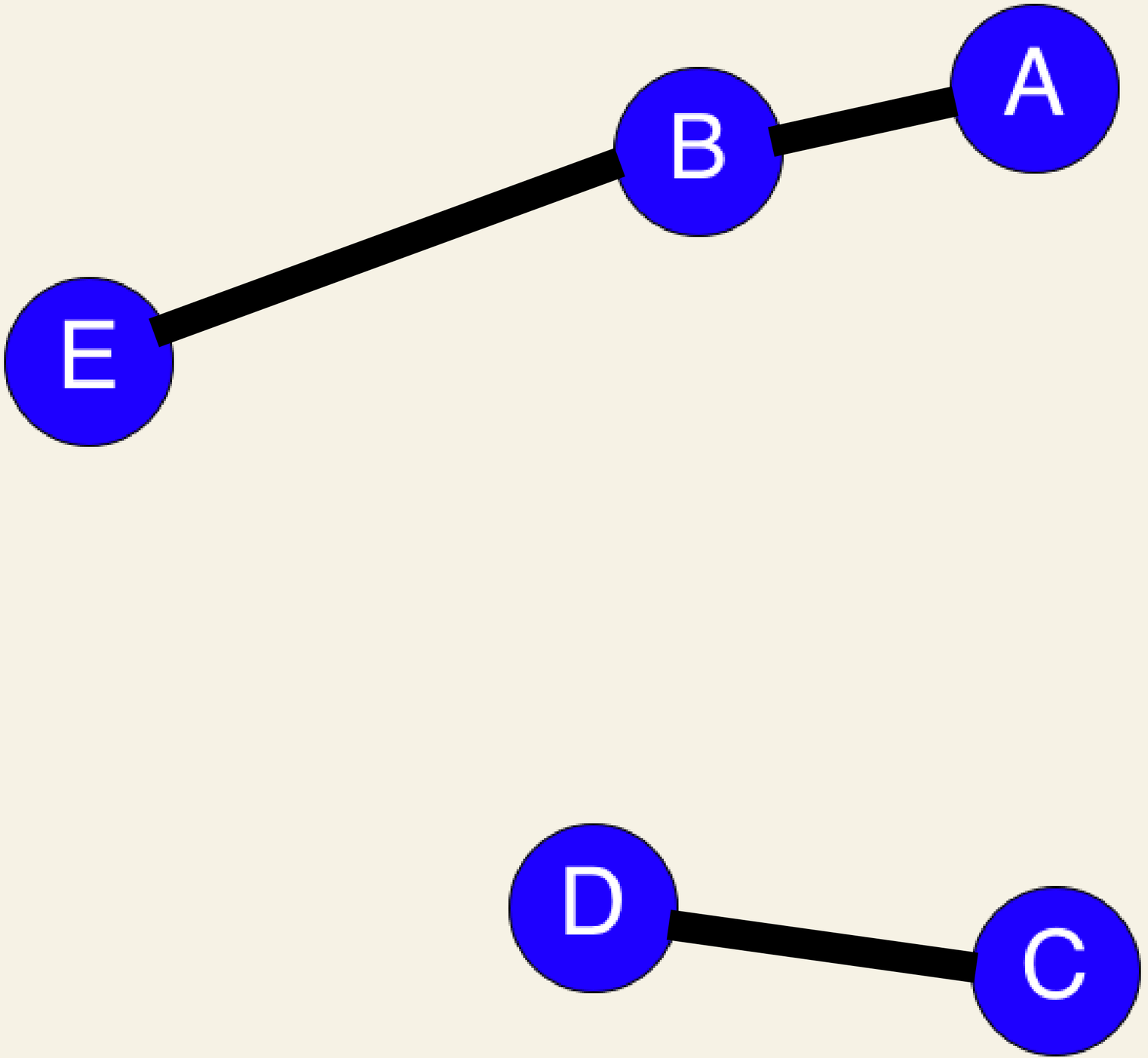
THE MULTIFRAGMENT HEURISTIC

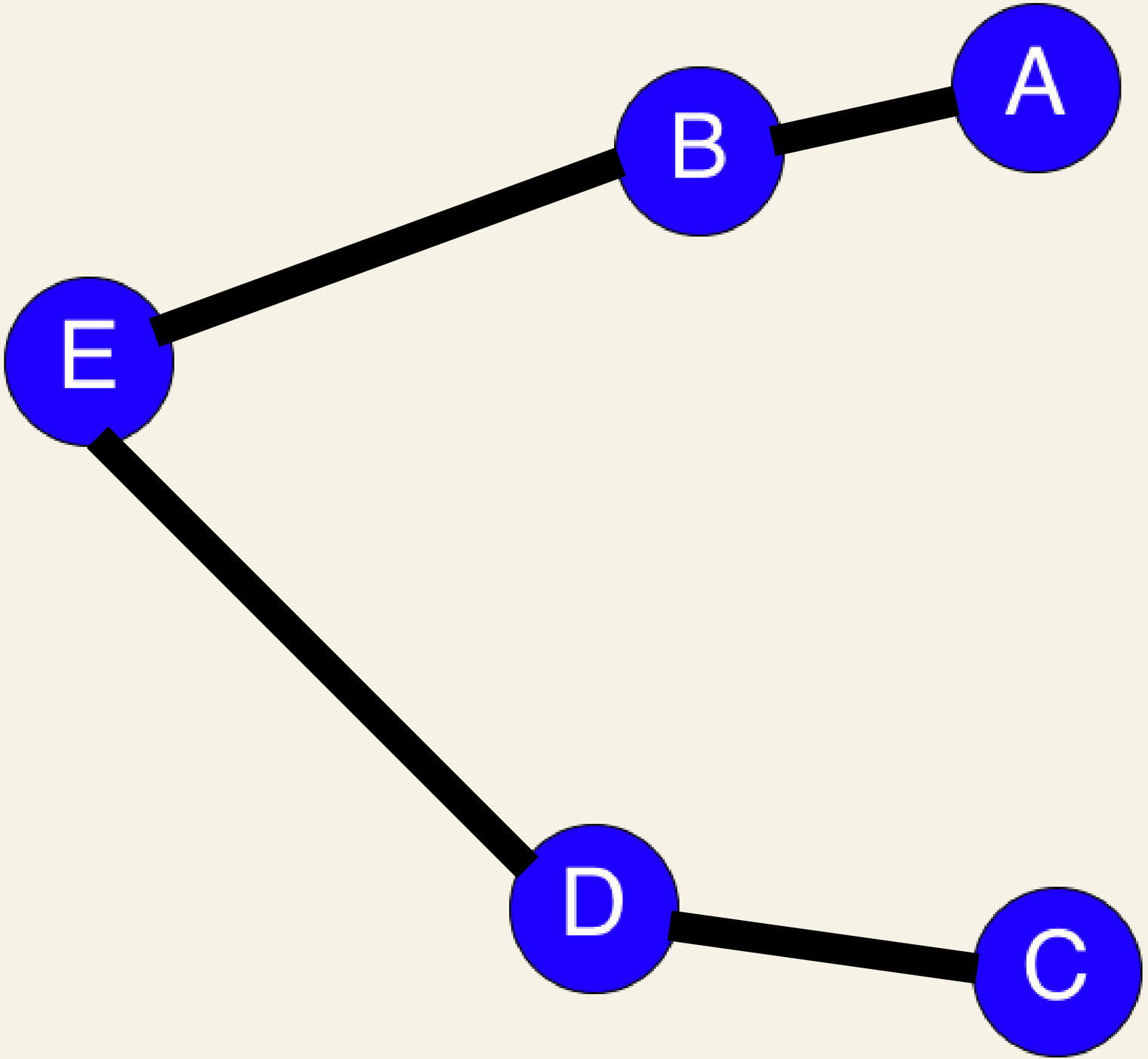
Intuition

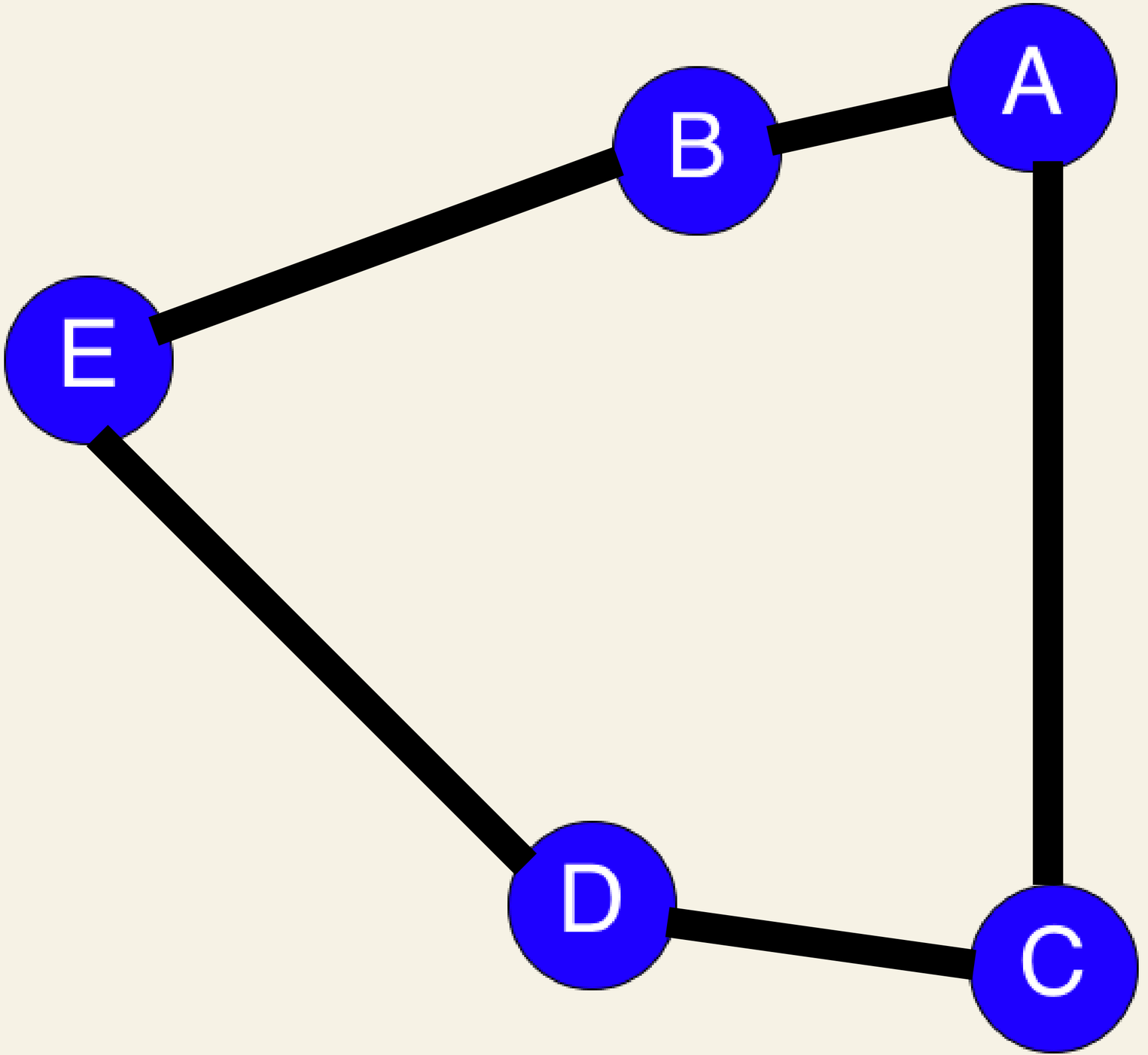

A
B
E
D
C

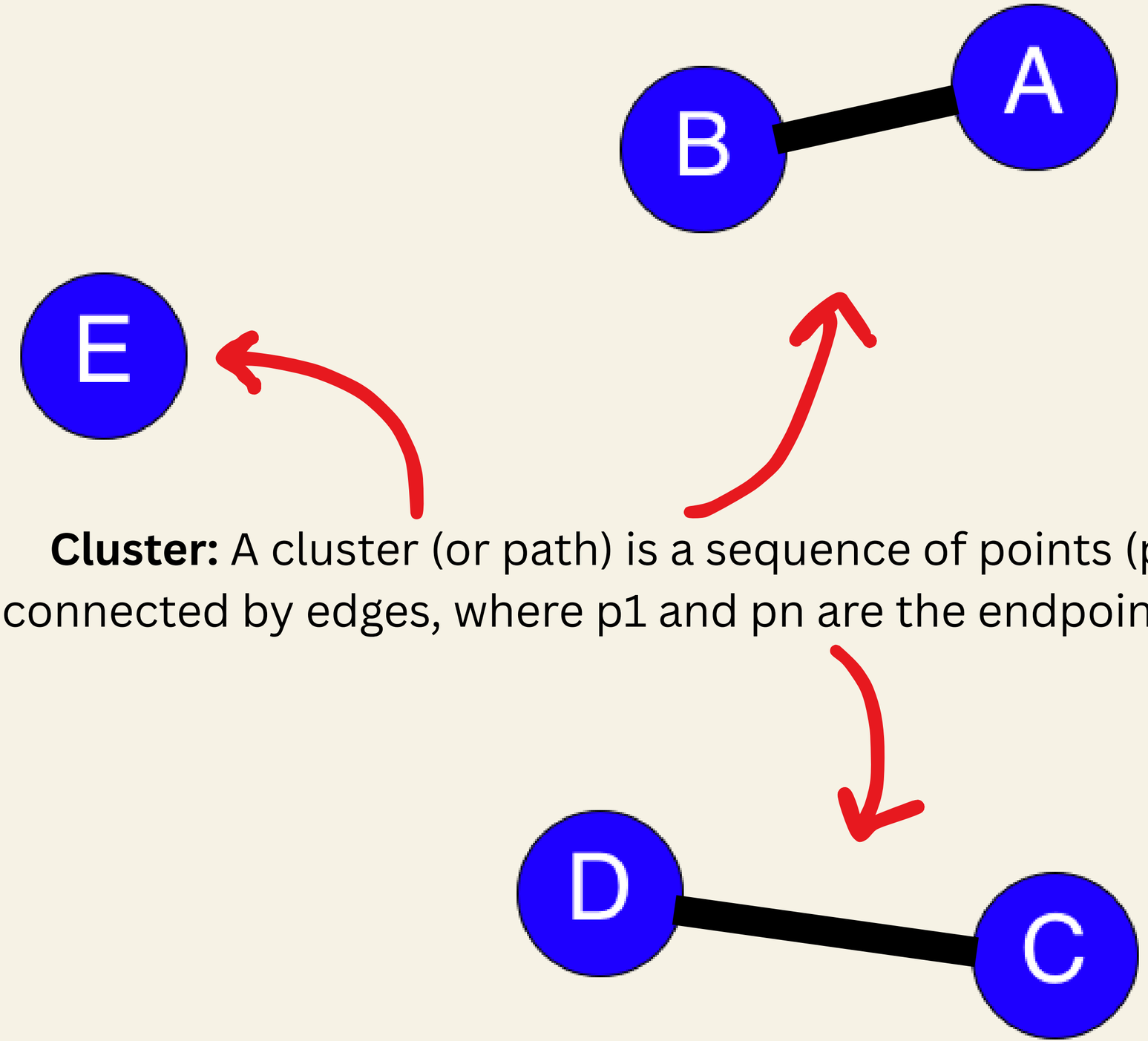

A
B
E
D
C


A
B
E
D
C

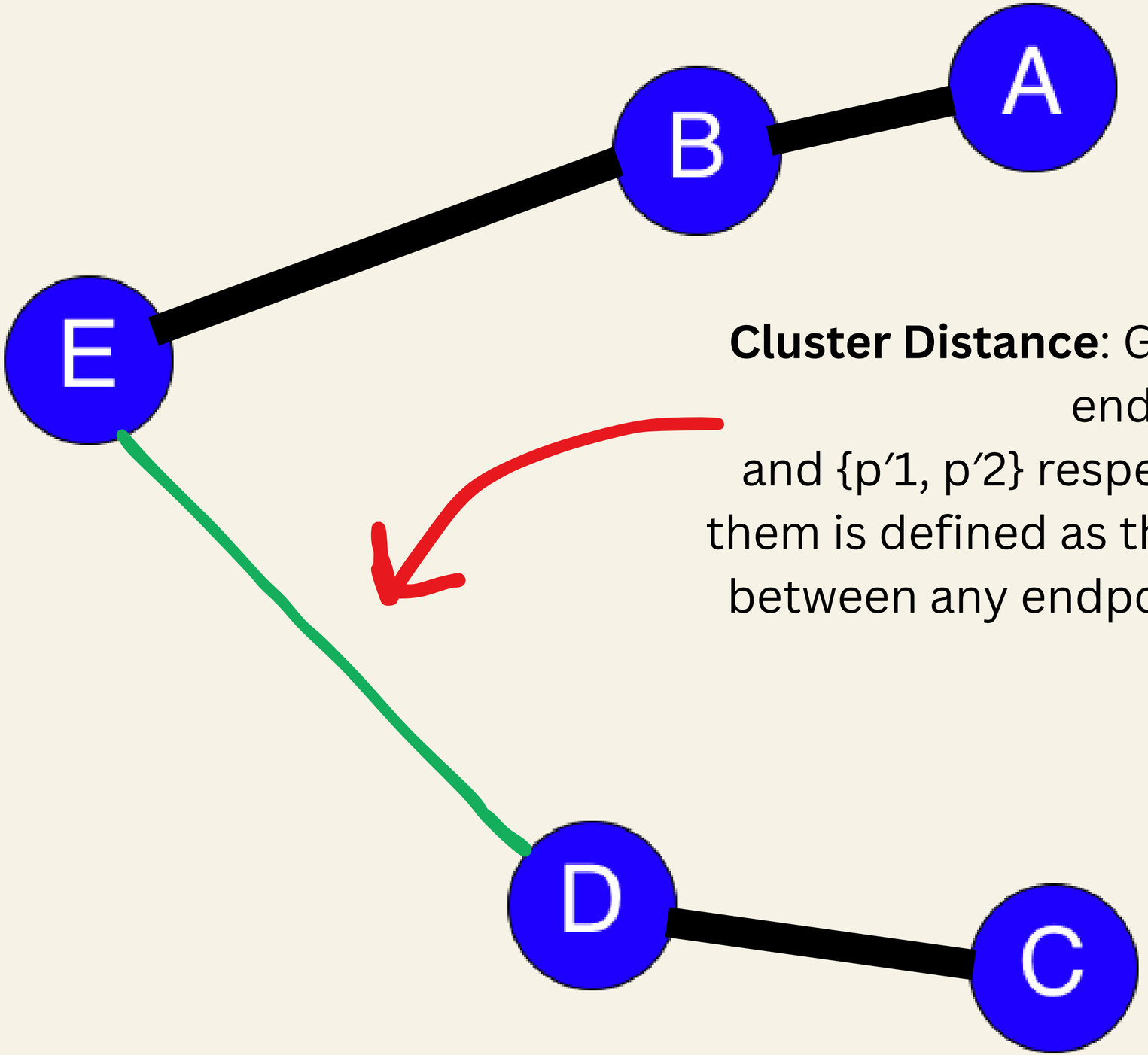

A
B
E
D
C


A
B
E
D
C


A
B
E
D
C

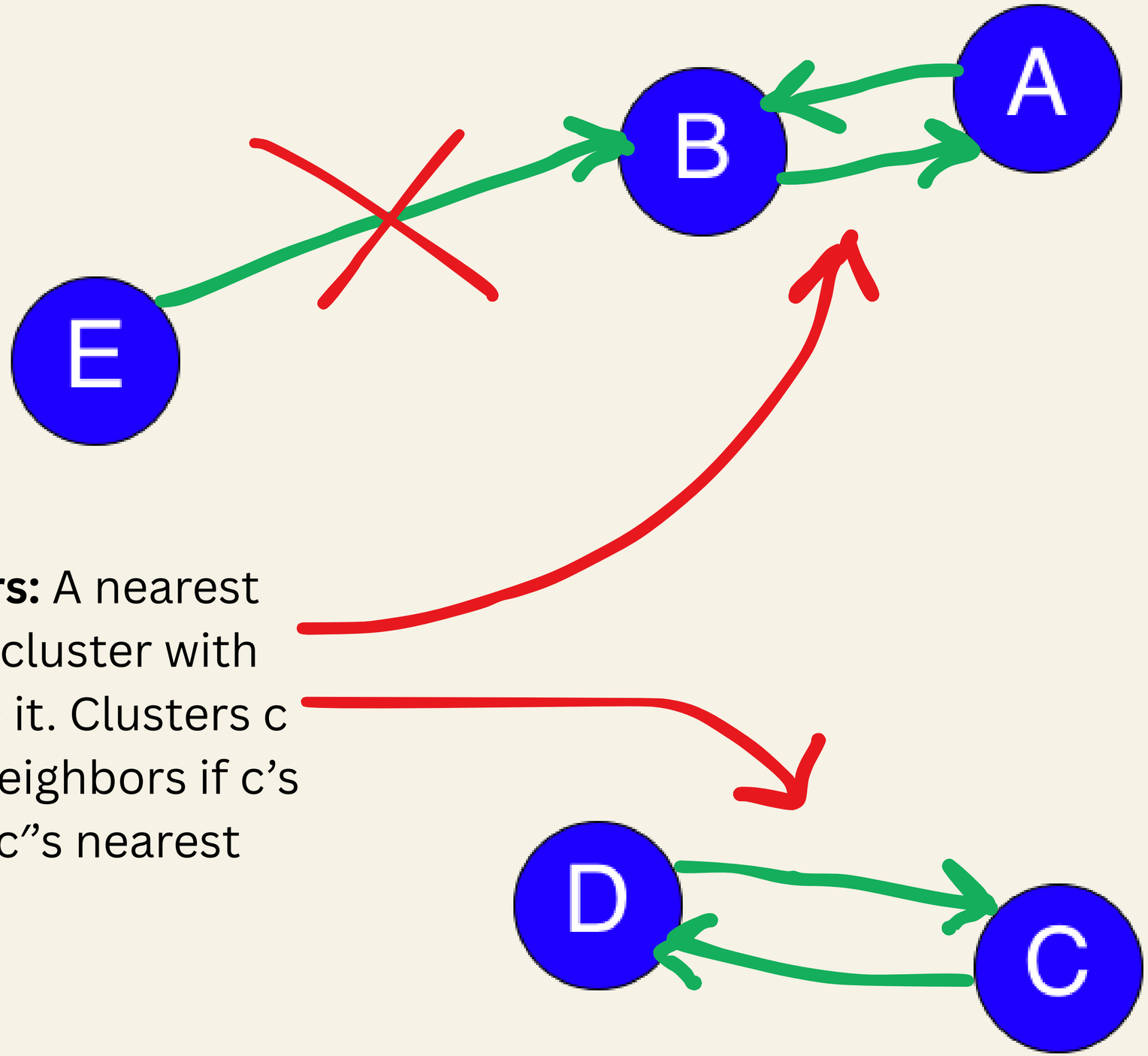


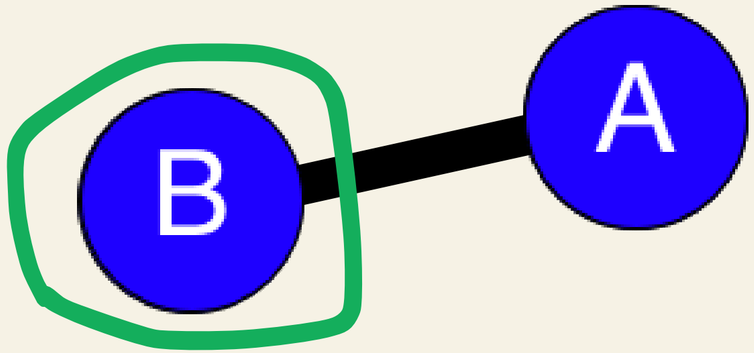
Cluster: A cluster (or path) is a sequence of points (p1, p2, . . . , pn) connected by edges, where p1 and pn are the endpoints of the cluster.



Cluster Distance: Given two clusters c and c′ with endpoints {p1, p2}
and {p′1, p′2} respectively, the distance between them is defined as the minimum Euclidean distance between any endpoint of c and any endpoint of c′

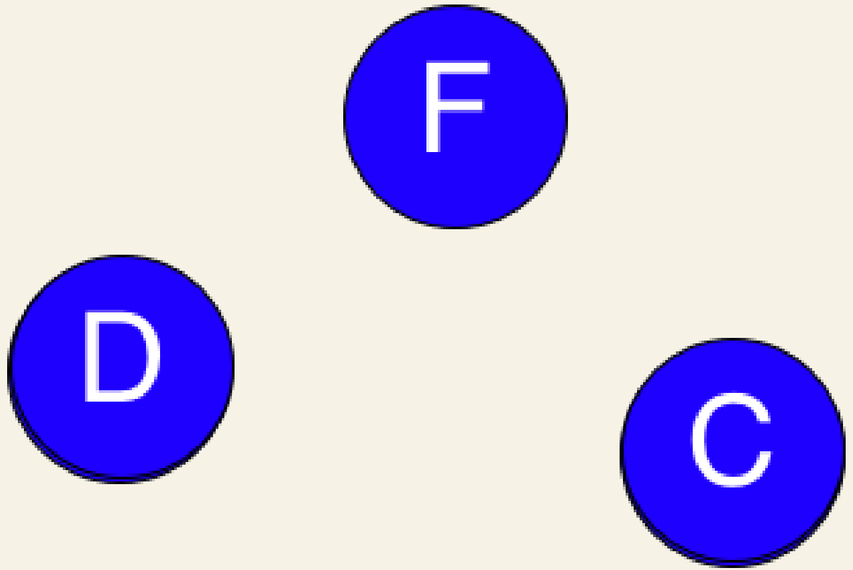
Mutually Nearest Neighbors: A nearest neighbor of a cluster is the cluster with minimum cluster distance to it. Clusters c and c′ are mutually nearest neighbors if c's nearest neighbor is c′ and c″s nearest neighbor is c.

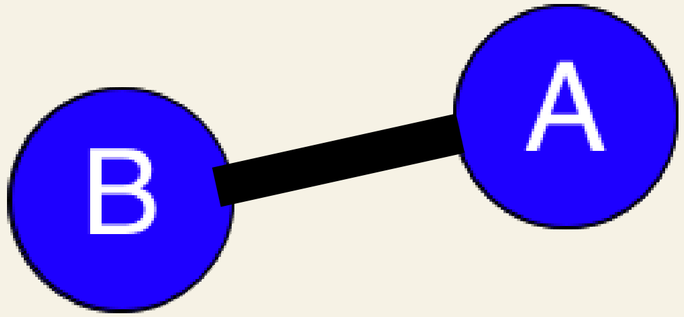


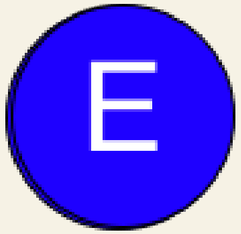


E

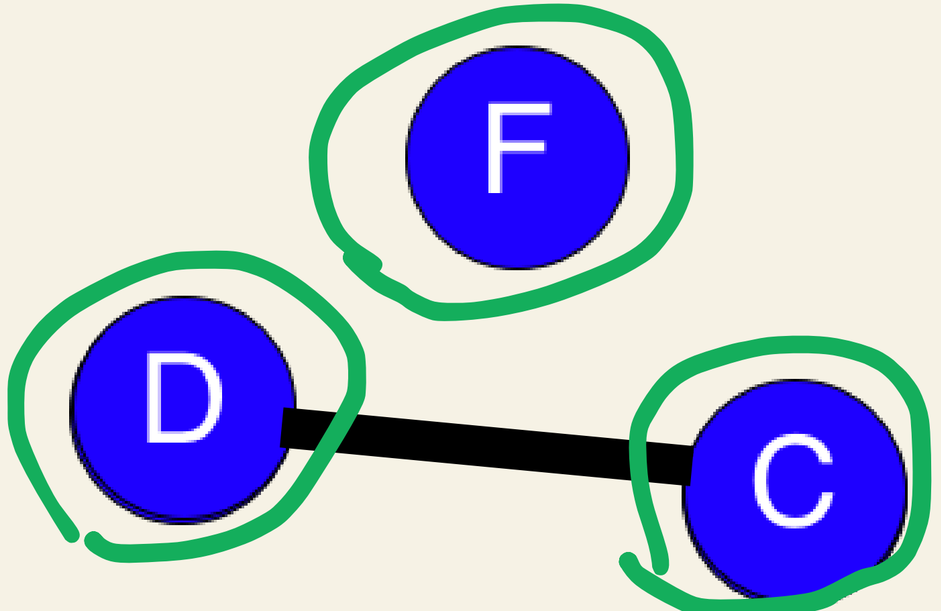
Hard answer: A single point that is the nearest neighbor of the input point.







Soft answer: Three points that are mutually closer to each other than the input point is to its nearest neighbor.



Algorithm Structure

- Multifragment
  - NextCluster
    - Naive Approach
    - Soft Answers (QueryCluster)
      - SNNQuery

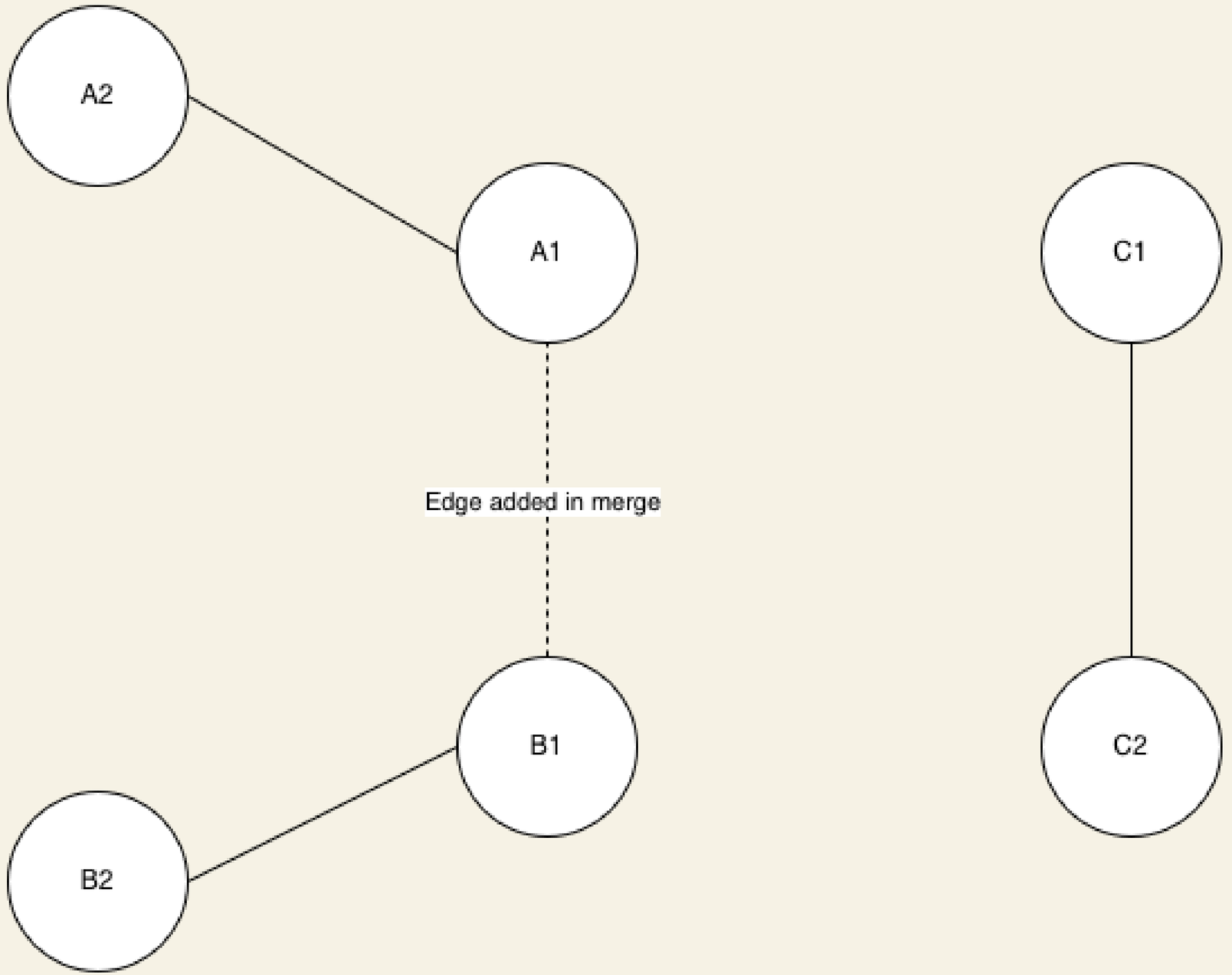
Global Local Equivalence

- Assuming that there are no ties in the pairwise distances between clusters, the two strategies for picking nearest neighbors produce the same tour.
- Requires **reducibility** and **monotonic decrease**

Reducibility

Lemma 6 (Reducibility) *Let A, B, and C be three distinct clusters, and let $A \cup B$ denote the cluster formed by merging A and B. Then*

$$d(A \cup B, C) \;\geq\; \min\big(d(A, C),\; d(B, C)\big).$$


A2
A1
C1
Edge added in merge
B1
C2
B2

Monotonic Decrease

Lemma 7 (Strict Monotonic Decrease along a Chain) *Let $c_1, c_2, \ldots, c_k$ be a chain, where c_{i+1} is the nearest neighbor of c_i for every i. Under the no-ties assumption,*

$$d(c_1, c_2) > d(c_2, c_3) > \cdots > d(c_{k-1}, c_k).$$

Chain Must End

$$d(A, B) > d(B, C) > d(C, A) >$$
$$d(A, B), \text{ which is a contradiction. } d(A, B)$$
$$\text{cannot be greater than } d(A, B).$$

Multifragment

Algorithm 2.1: MULTIFRAGMENT

Require: Point set P with $|P| = n$, neighbor count k, approximation factor ε
Ensure: Tour edge set E

$\mathcal{C} \leftarrow \{\, \{p\} : p \in P \,\}$
$E \leftarrow \emptyset$; $S \leftarrow$ empty stack
while $|\mathcal{C}| > 1$ **do**
    if S is empty **then**
        push $(c, \perp)$ onto S for an arbitrary $c \in \mathcal{C}$
        continue
    $(u, v) \leftarrow \text{top}(S)$
    $(a, b) \leftarrow \text{NEXTCLUSTER}(u, v)$
    if $v \neq \perp$ **and** $\{a, b\} = \{u, v\}$ **then**
        pop S
        if S nonempty **and** top of S contains u or v **then**
            pop S
        merge u and v into a new cluster in $\mathcal{C}$, add the connecting edge to E
    else
        push (a, b) onto S
add the closing edge between the endpoints of the final path to E
return E

Multifragment

Table 2.1: *Trace of the multifragment heuristic on Figure 2.1.*

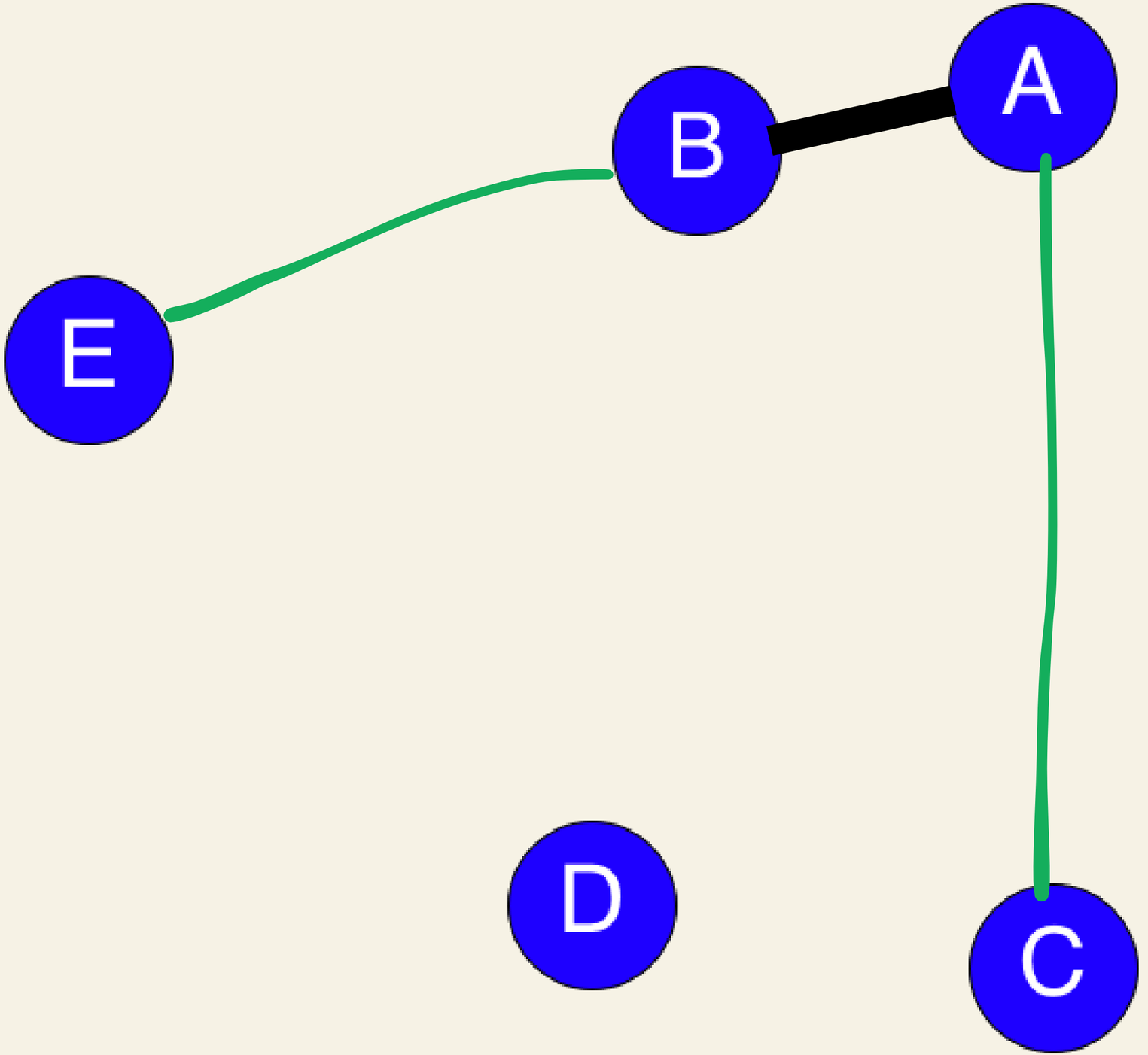
Step	Action	Stack
0	initialize	$[\,]$
1	push $(A, \perp)$	$[(A, \perp)]$
2	$\mathrm{NN}(A) = B$; push (A, B)	$[(A, \perp), (A, B)]$
3	$\mathrm{NN}(B) = A$: merge AB; add edge AB	$[\,]$
4	push $(C, \perp)$	$[(C, \perp)]$
5	$\mathrm{NN}(C) = D$; push (C, D)	$[(C, \perp), (C, D)]$
6	$\mathrm{NN}(D) = C$: merge CD; add edge CD	$[\,]$
7	push $(E, \perp)$	$[(E, \perp)]$
8	$\mathrm{NN}(E) = AB$; push (E, AB)	$[(E, \perp), (E, AB)]$
9	$\mathrm{NN}(AB) = E$: merge ABE; add edge BE	$[\,]$
10	push $(ABE, \perp)$	$[(ABE, \perp)]$
11	$\mathrm{NN}(ABE) = CD$; push (ABE, CD)	$[(ABE, \perp), (ABE, CD)]$
12	$\mathrm{NN}(CD) = ABE$: merge; add edge ED	$[\,]$
13	close: add edge AC	$[\,]$

NextCluster

Algorithm 2.2: NextCluster *(naive variant, hard answers only)*

Require: Cluster pair (u, v), where v may be $\perp$
Ensure: Candidate cluster pair (a, b)

if $v = \perp$ **then**
    $c \leftarrow u$
else
    $c \leftarrow v$
$(q_1, q_2) \leftarrow$ endpoints of c
$p_1 \leftarrow$ exact nearest neighbor of q_1 in (active points $\setminus \{q_1, q_2\}$)
$p_2 \leftarrow$ exact nearest neighbor of q_2 in (active points $\setminus \{q_1, q_2\}$)
$p^* \leftarrow$ whichever of p_1, p_2 is closer to its query point
return $(c,\ \text{cluster}(p^*))$


A
B
E
D
C

NextCluster

Algorithm 2.3: NEXTCLUSTER *(soft variant)*

Require: Cluster pair (u, v), where v may be $\perp$
Ensure: Candidate cluster pair (a, b)

1: $M \leftarrow \{u\} \cup \text{QUERYCLUSTER}(u)$
2: **if** $v \neq \perp$ **then**
3: $\quad M \leftarrow M \cup \{v\} \cup \text{QUERYCLUSTER}(v)$
4: **return** $\arg\min_{\{x,y\} \subseteq M} d(x, y)$

QueryCluster

Algorithm 2.4: QUERYCLUSTER

Require: Cluster c with endpoints q_1, q_2, active point set P, neighbor count k, approximation factor ε

Ensure: Set $\mathcal{S}$ of candidate cluster IDs

1: $A_1 \leftarrow \textsc{SnnQuery}(q_1,\ P \setminus \{q_1, q_2\},\ k,\ \varepsilon)$

2: $A_2 \leftarrow \textsc{SnnQuery}(q_2,\ P \setminus \{q_1, q_2\},\ k,\ \varepsilon)$

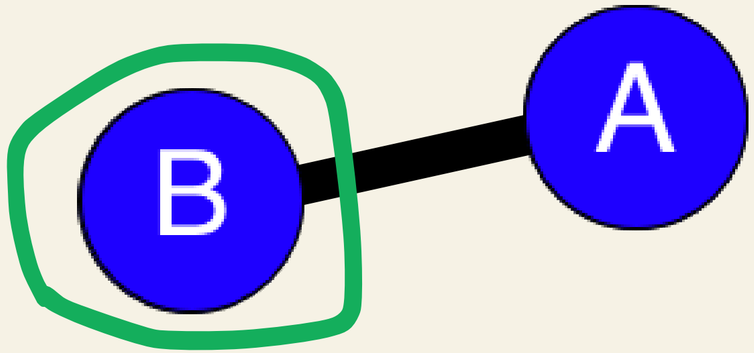
3: **return** $\{\, \text{cluster}(p) : p \in A_1 \cup A_2,\ \text{cluster}(p) \neq c \,\}$

SNNQuery

Algorithm 2.5: SnnQuery

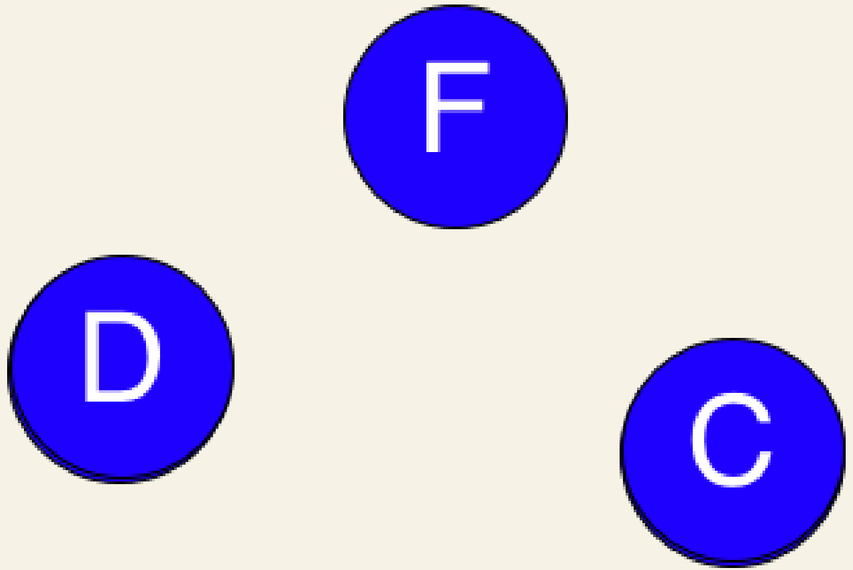
Require: Query point q, point set P, neighbor count k, approximation factor ε
Ensure: A hard answer $p^* \in P$, or a soft answer $\{a, b, c\} \subseteq P$ with pairwise distances all less than $d(q, p^*)$

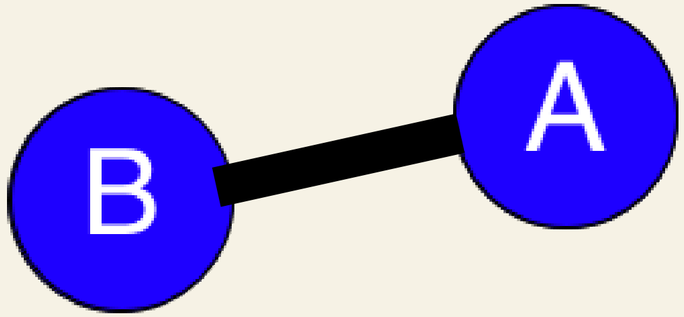
$N \leftarrow (1+\varepsilon)$-approximate k nearest neighbors of q in P
$p^* \leftarrow \arg\min_{p \in N} d(q, p)$
$d^* \leftarrow d(q, p^*)$
for all triples $\{a, b, c\} \subseteq N$ **do**
    if $d(a, b) < d^*$ **and** $d(a, c) < d^*$ **and** $d(b, c) < d^*$ **then**
        return soft answer $\{a, b, c\}$
return hard answer p^*

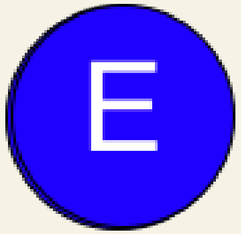


E

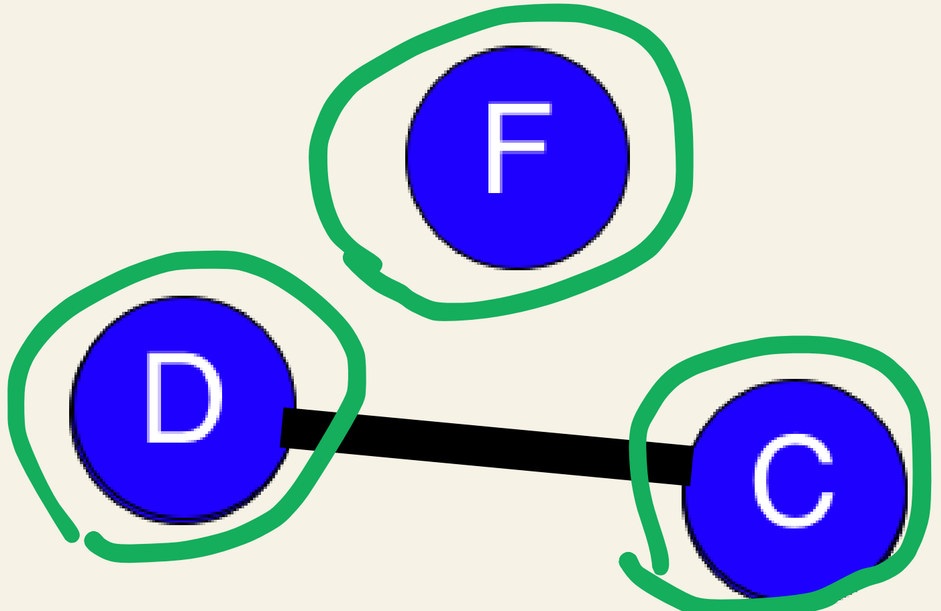
Hard answer: A single point that is the nearest neighbor of the input point.





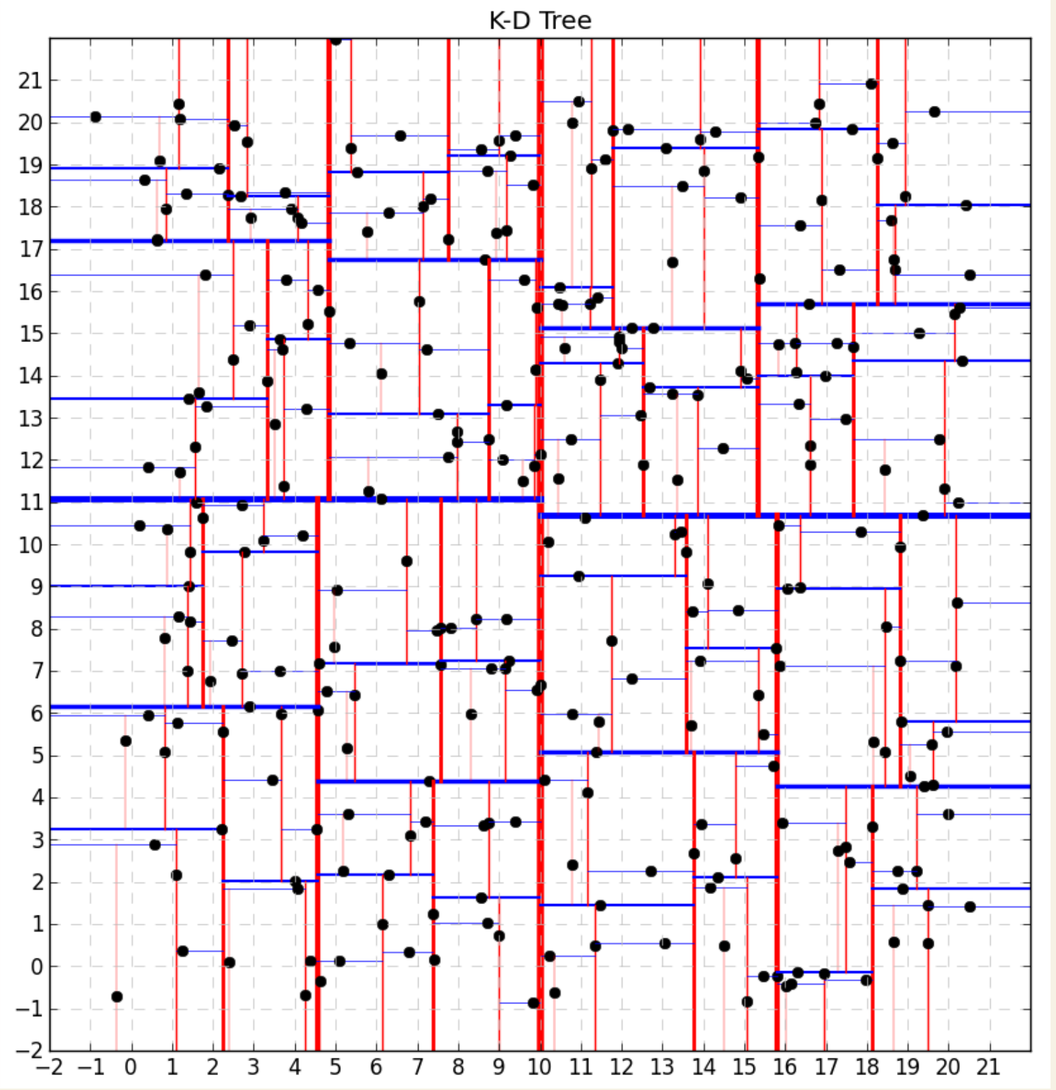


Soft answer: Three points that are mutually closer to each other than the input point is to its nearest neighbor.



ANN Data Structure

- KD Tree (split nodes)
- BD Tree (shrink nodes)
- k and ε



2-APPROXIMATION

4 Steps

- Minimum Spanning Tree (Prim’s Algorithm)
- Double edges
- Eulerian Circuit (Hierholzer’s Algorithm)
- Shortcut for Hamiltonian Tour

MINIMUM SPANNING TREE
PRIM'S ALGORITHM

Minimum Spanning Tree

- A **minimum spanning tree** is a subset of edges in a graph that connects all vertices together without creating any cycles and with minimum total weight.

Distances

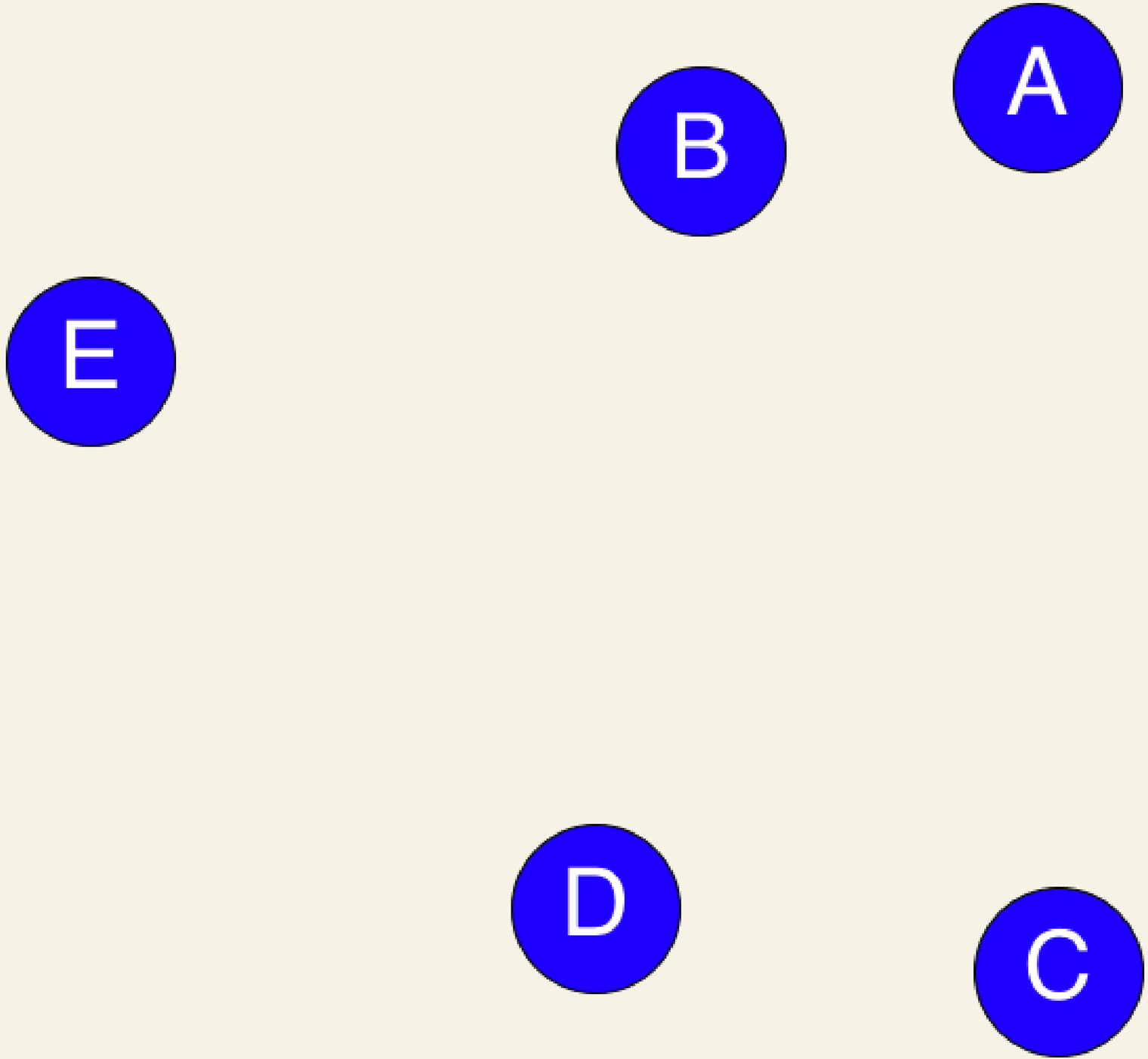
A: ∞

B: ∞

C: ∞

D: ∞

E: 0



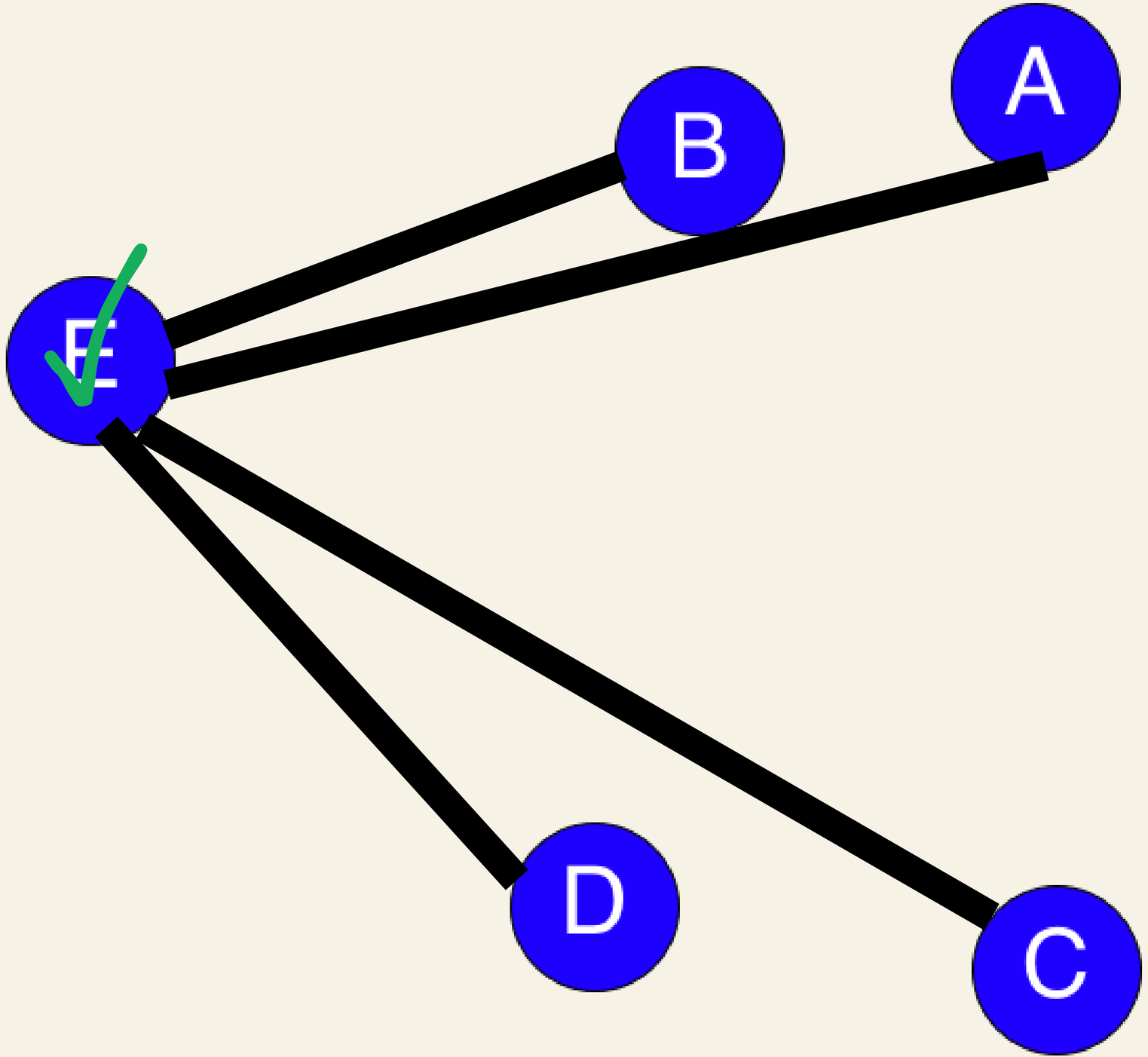
Distances

A: 6

B: 4

C: 7

D: 5



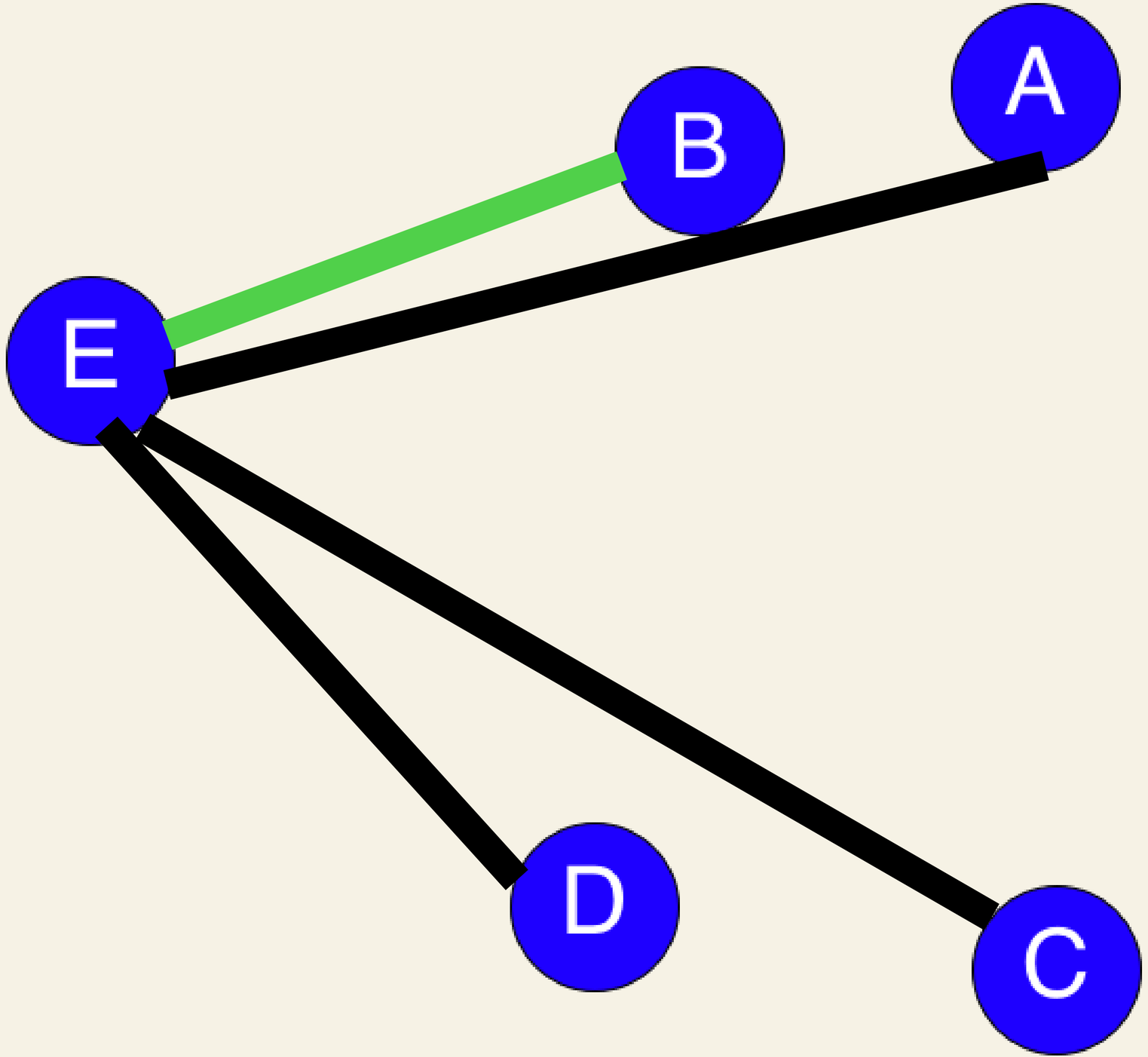
Distances

A: 6

B: 4

C: 7

D: 5



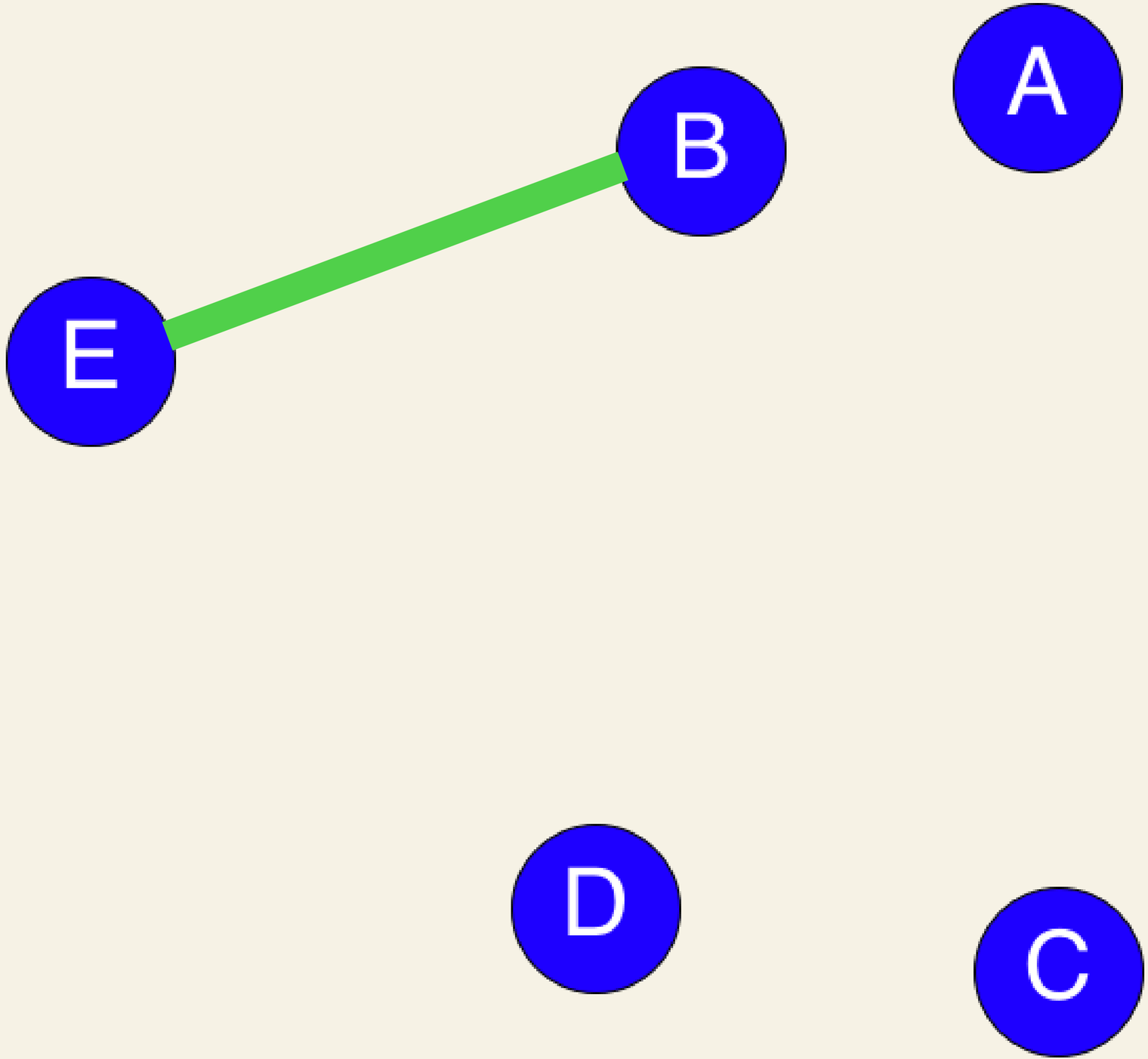
Distances

A: 6

B: 4

C: 7

D: 5

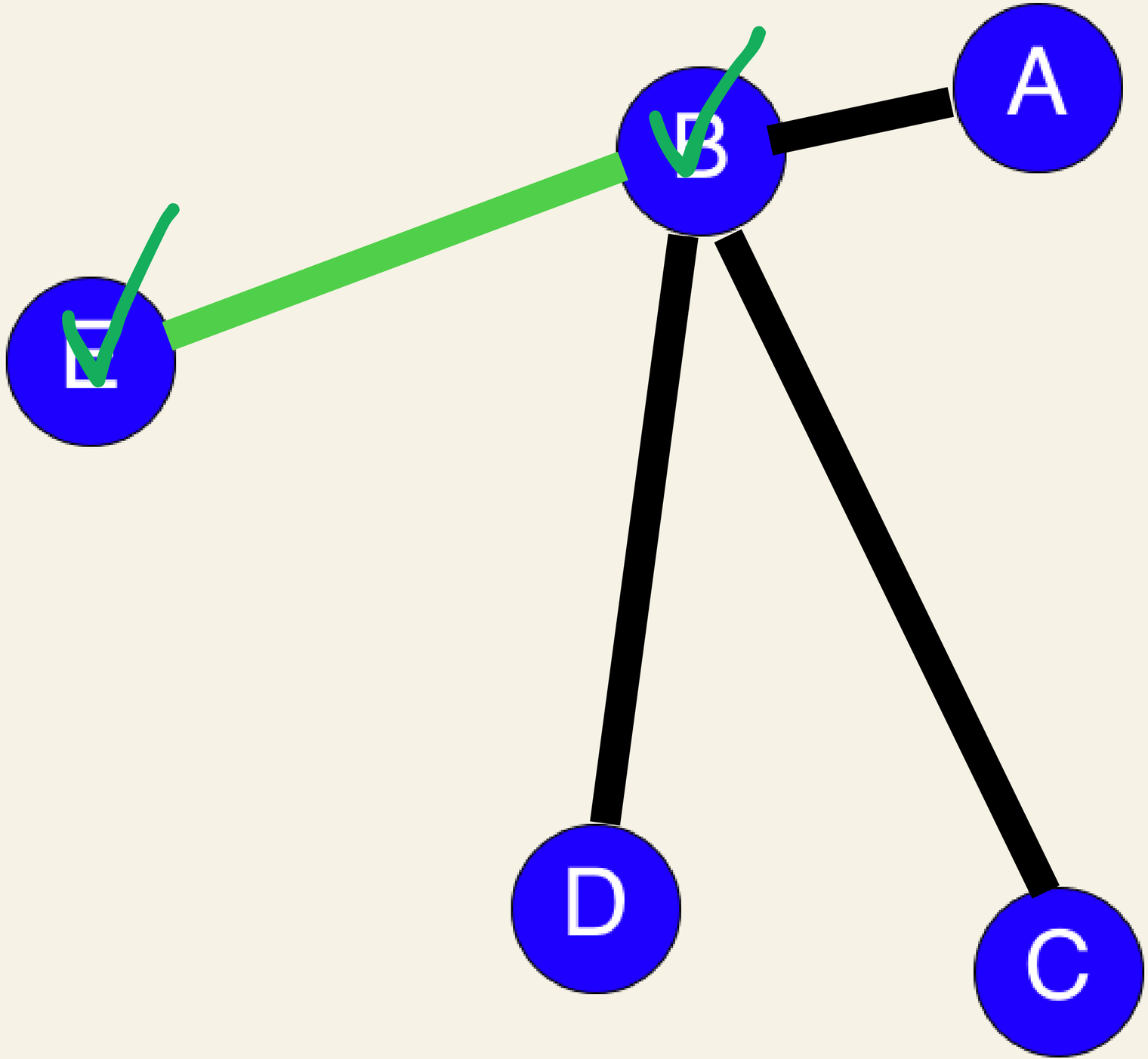


Distances

A: 2

C: 6.5

D: 5 (from E)

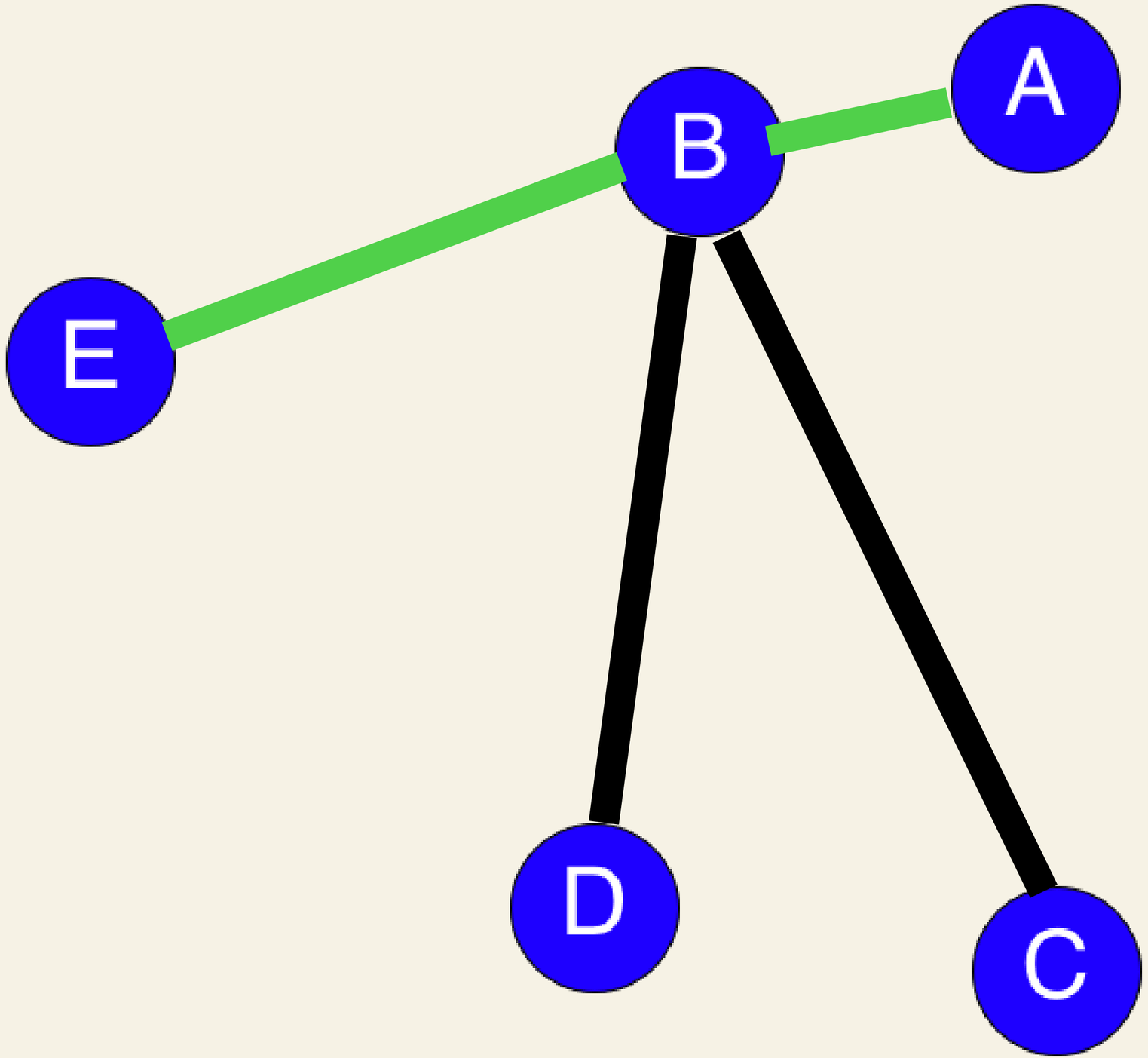


Distances

A: 2

C: 6.5

D: 5 (from E)

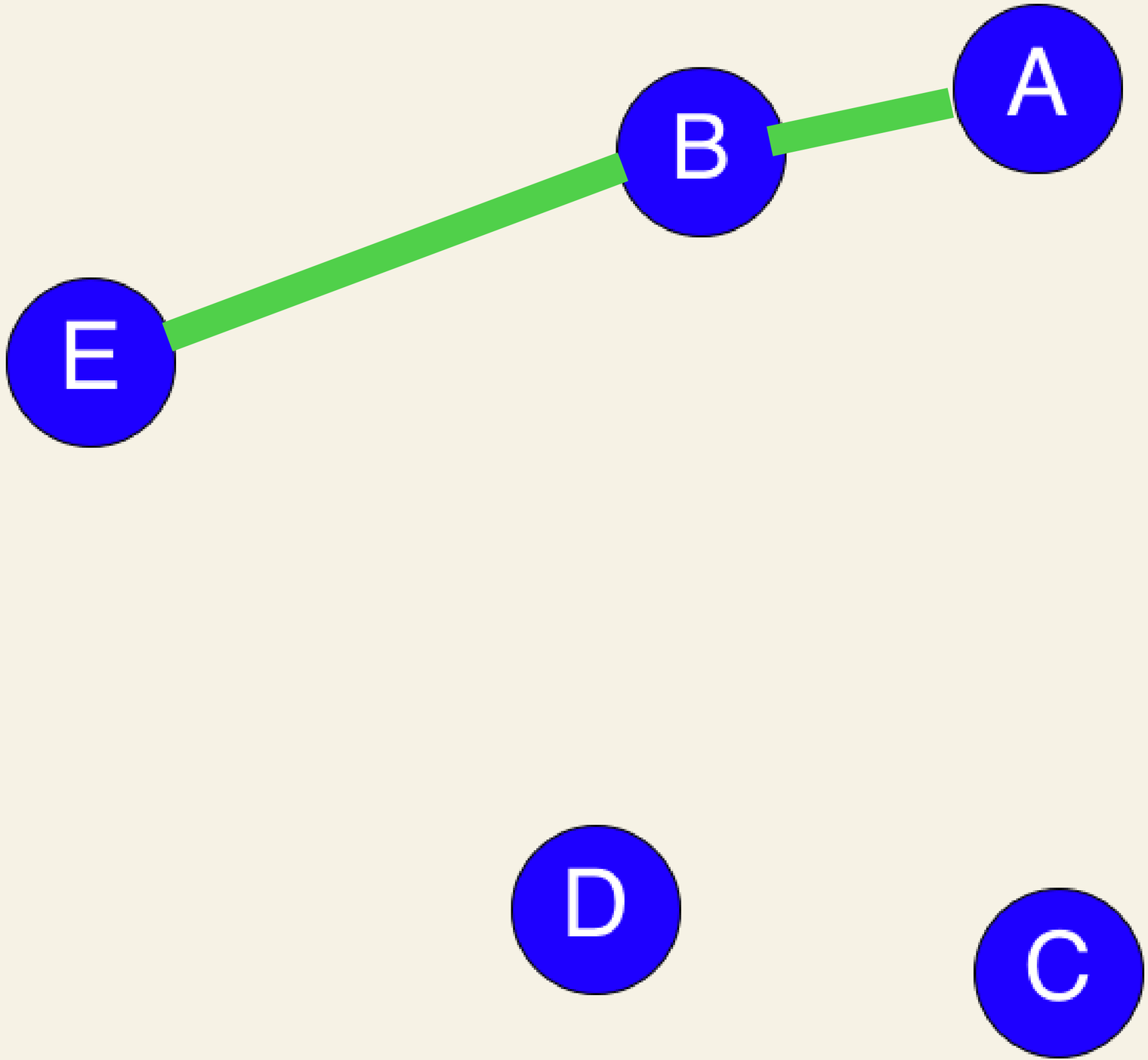


Distances

A: 2

C: 6.5

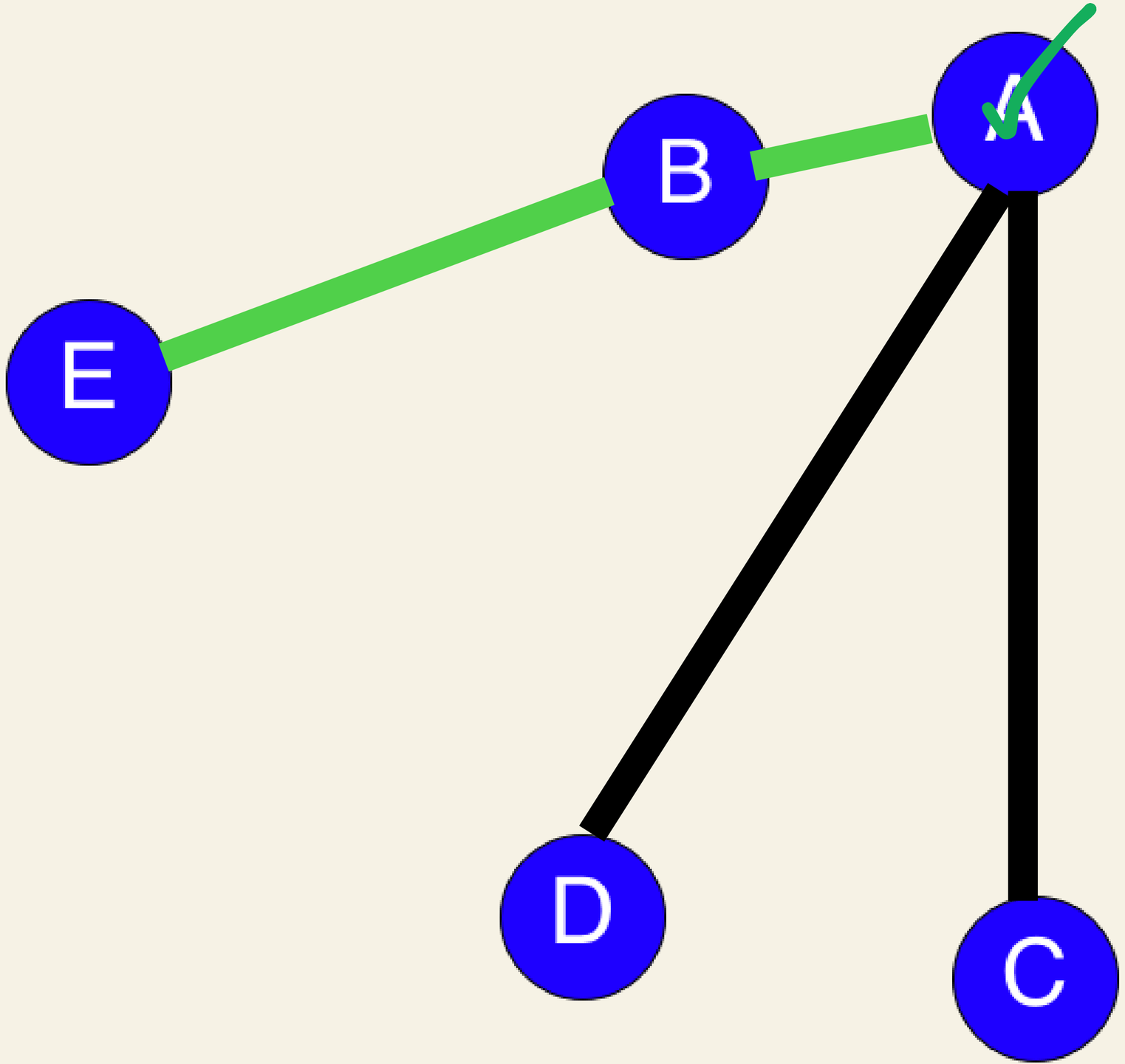
D: 5 (from E)



Distances

C: 6.5 (from B)

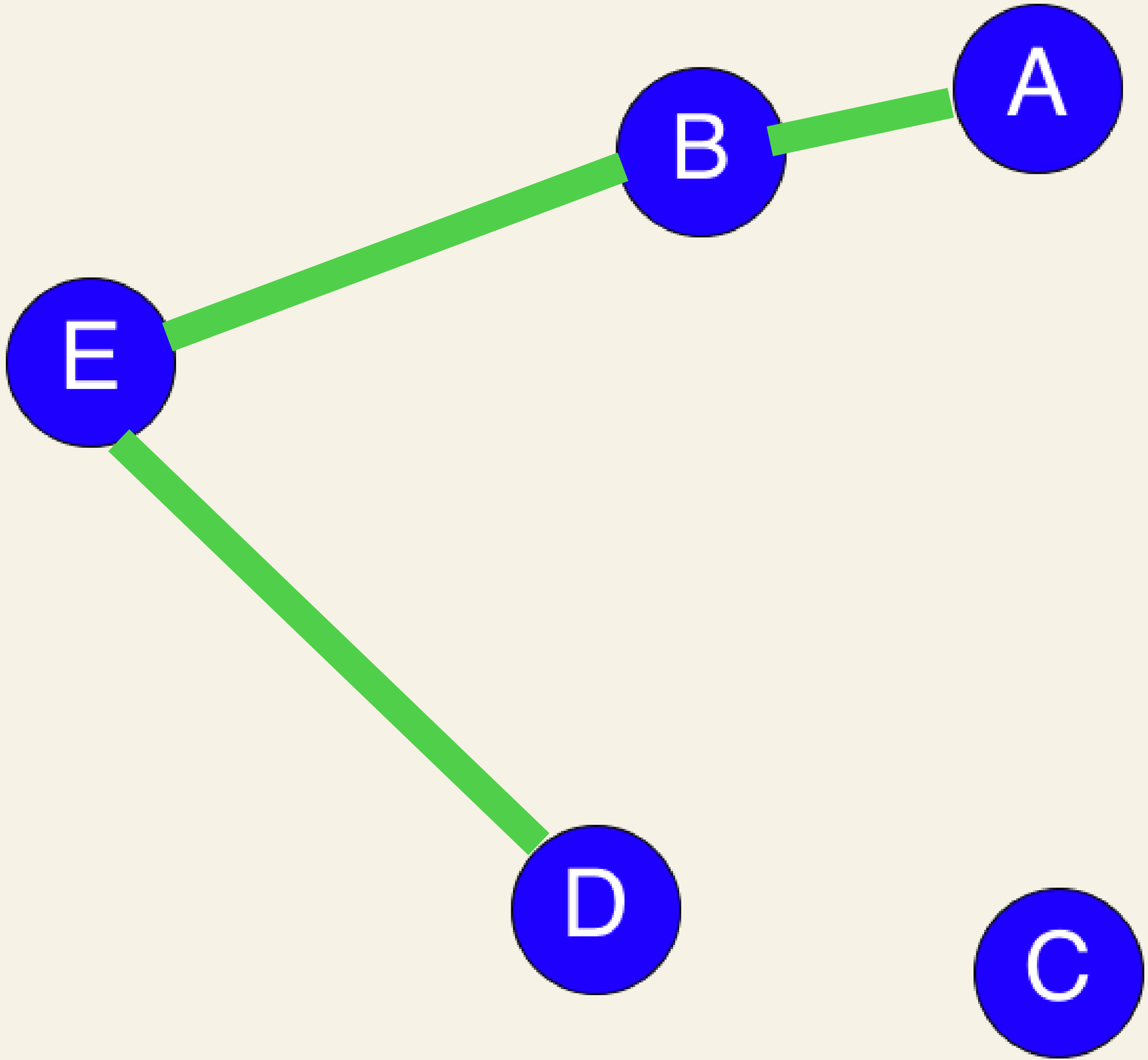
D: 5 (from E)



Distances

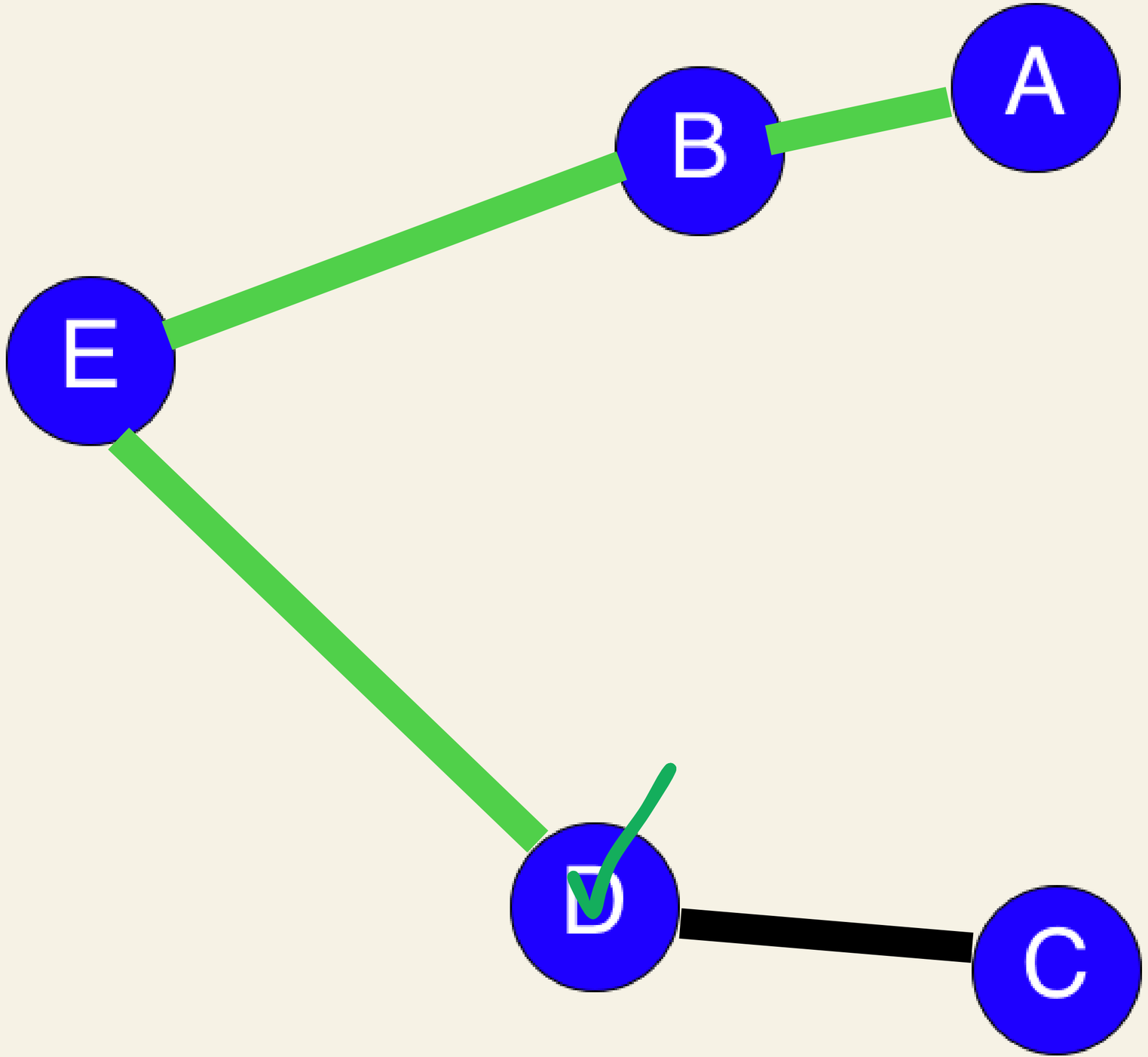
C: 6.5 (from B)

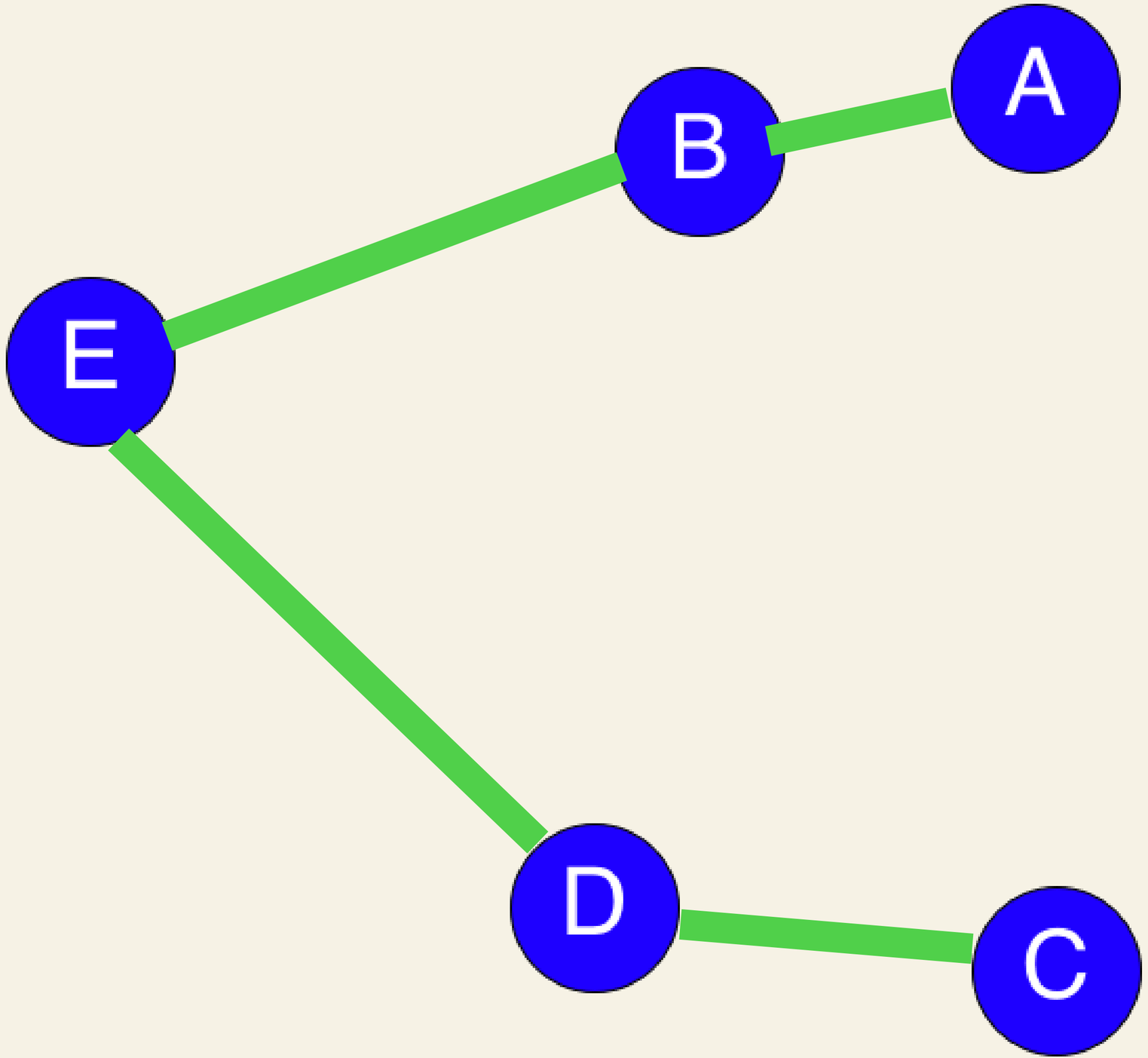
D: 5 (from E)



Distances

C: 2.5




A
B
E
D
C

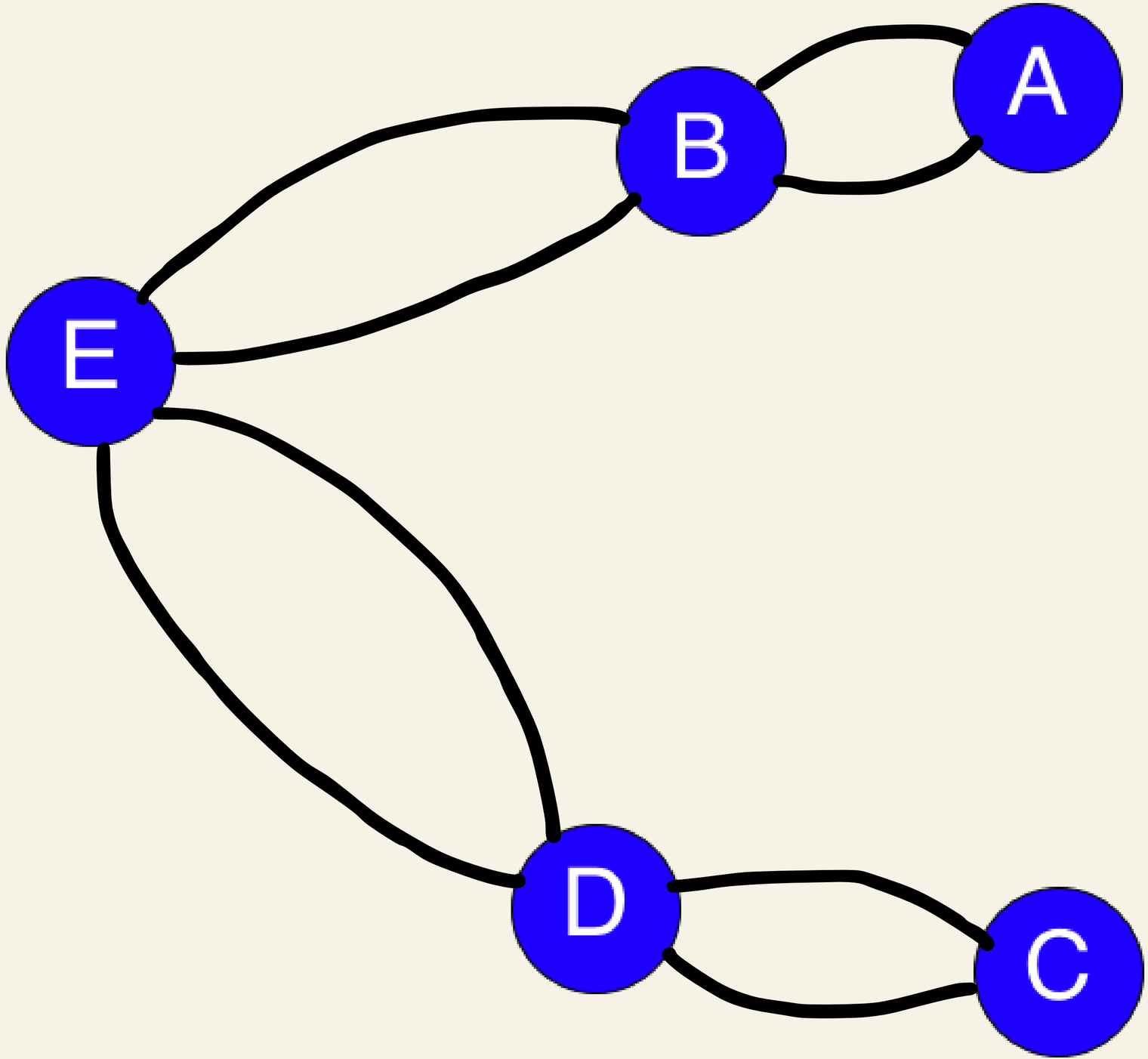
Complexity

- $O(n^2)$
- Bounds the whole algorithm
- MST is OPT for approximation

DOUBLING THE EDGES

Why do this?

- Even degree requirement
- Minimum Weight Perfect Matching on odd degree vertices


A
B
E
D
C

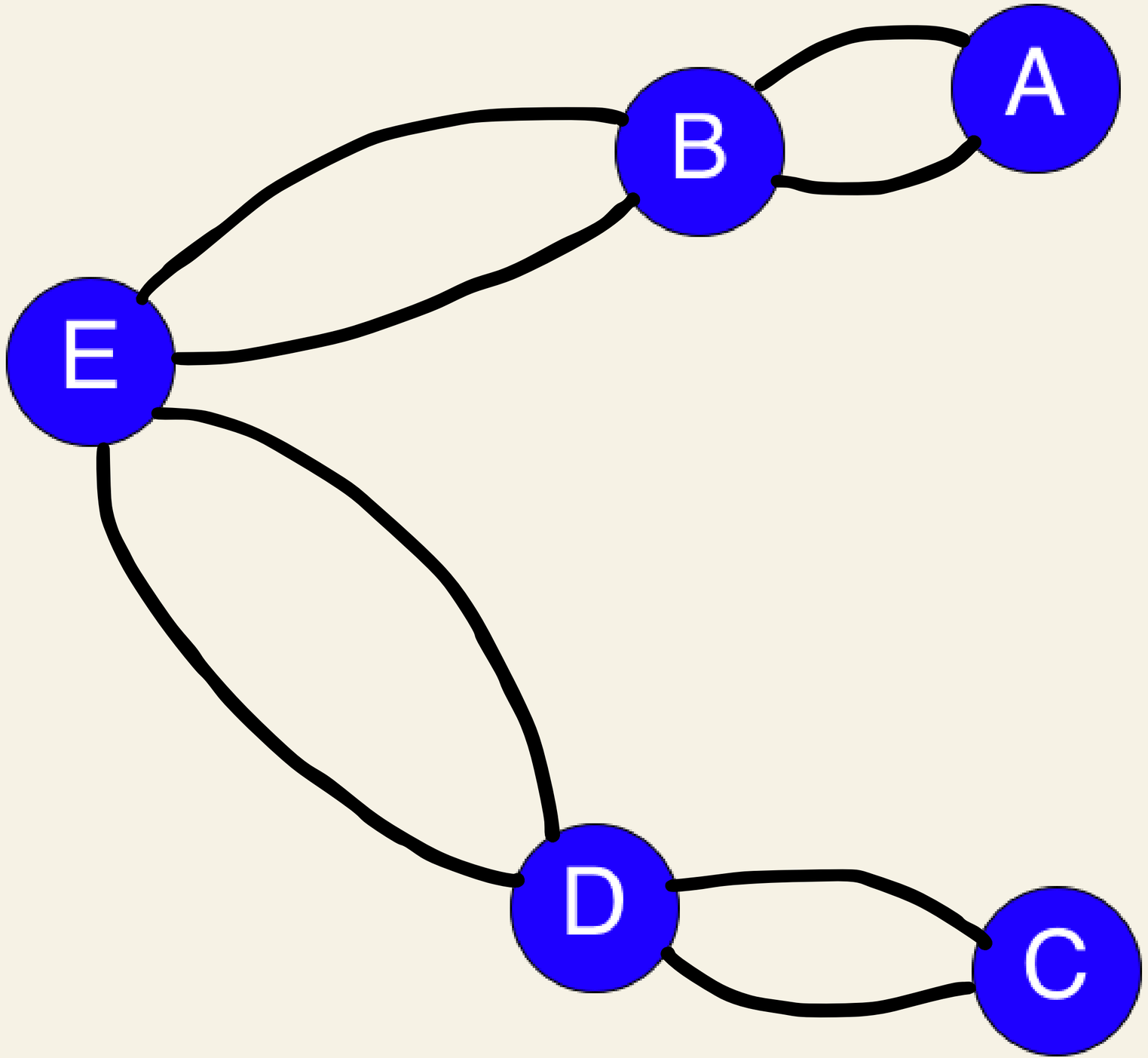
EULERIAN CIRCUIT
HIERHOLZER'S ALGORITHM

Eulerian Circuit

- An Eulerian circuit is a circuit that visits every edge in the graph exactly once and returns back to a starting vertex.

Stack

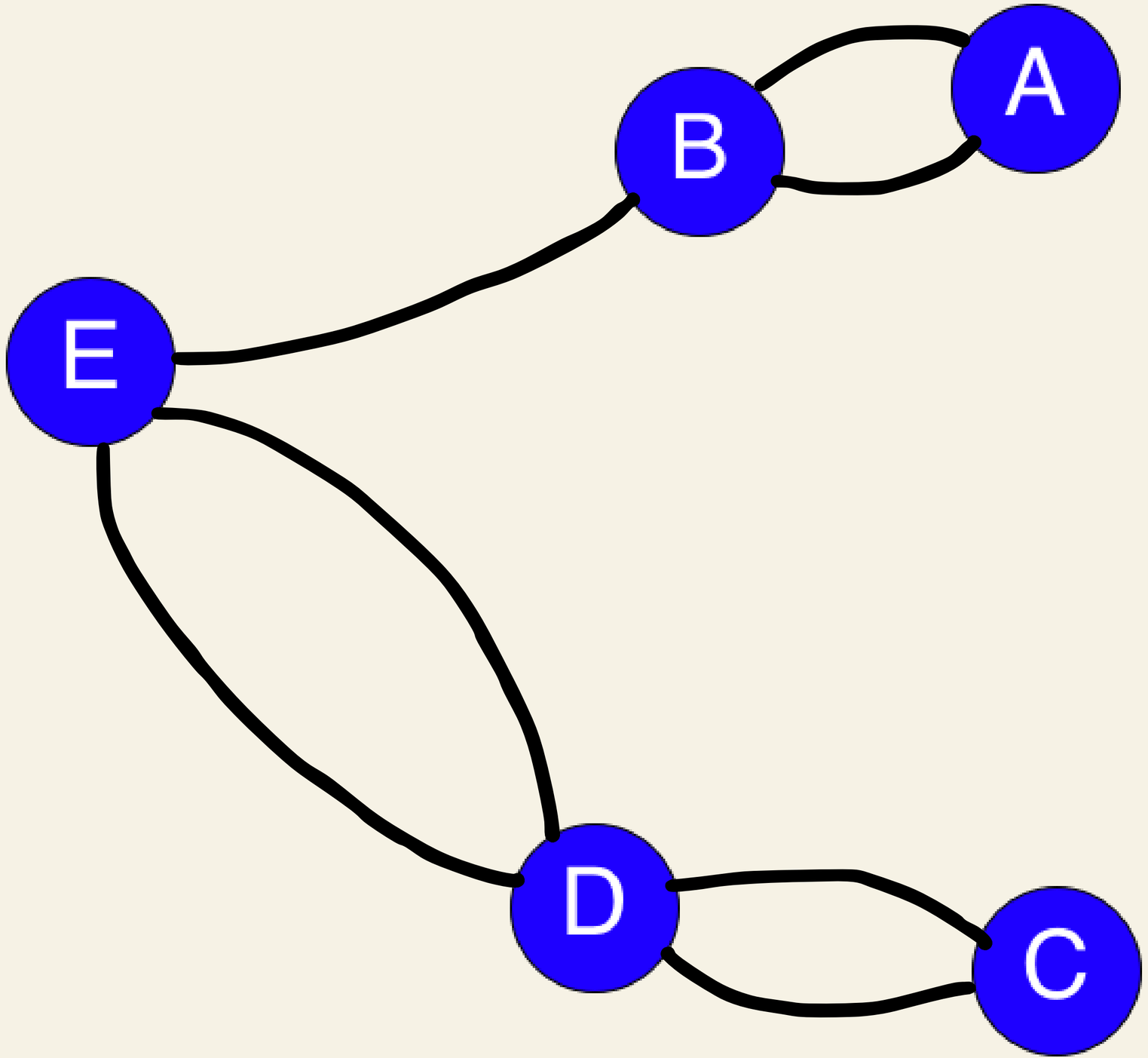
E



Stack

B

E

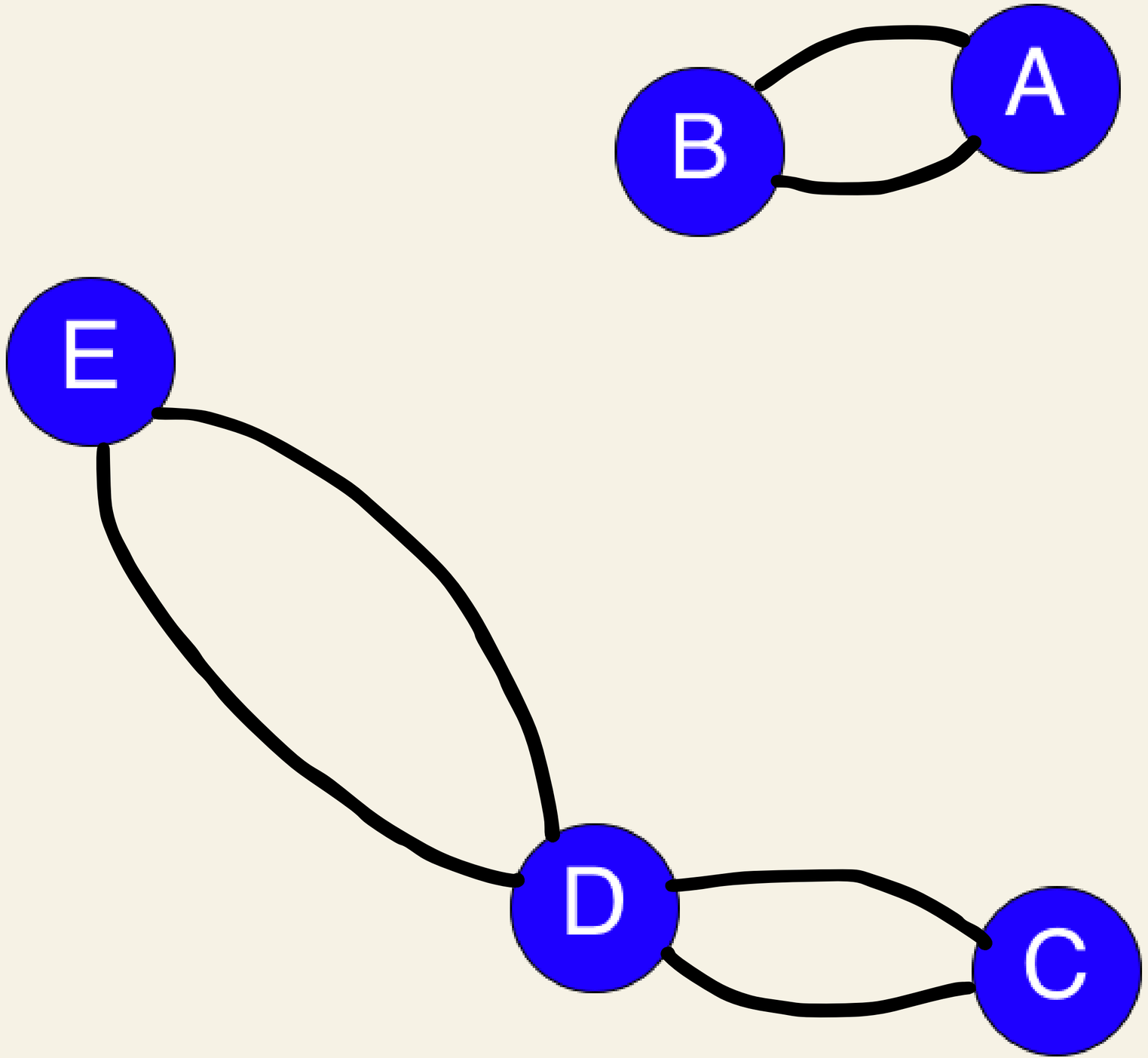


Stack

E

B

E



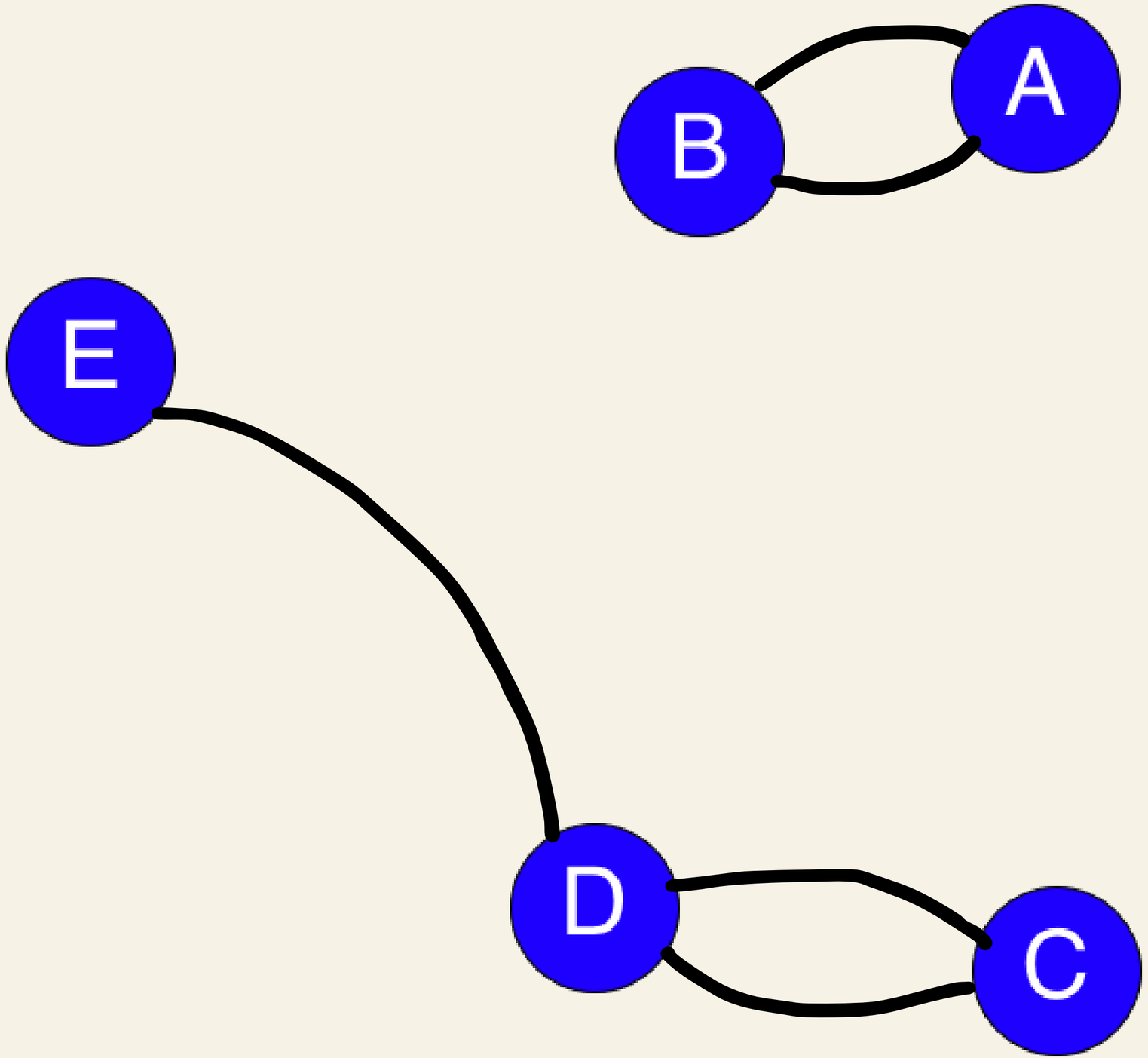
Stack

D

E

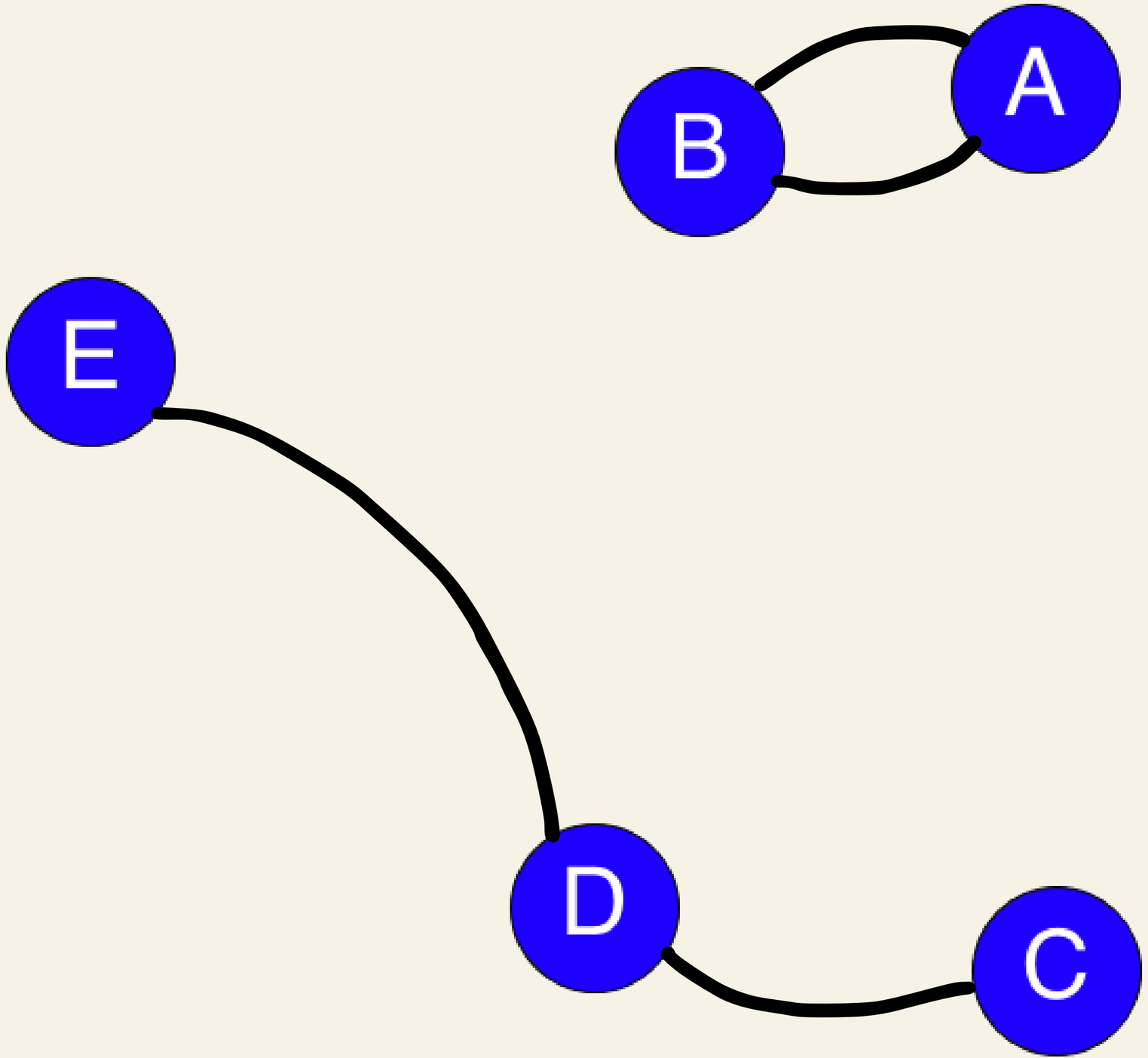
B

E



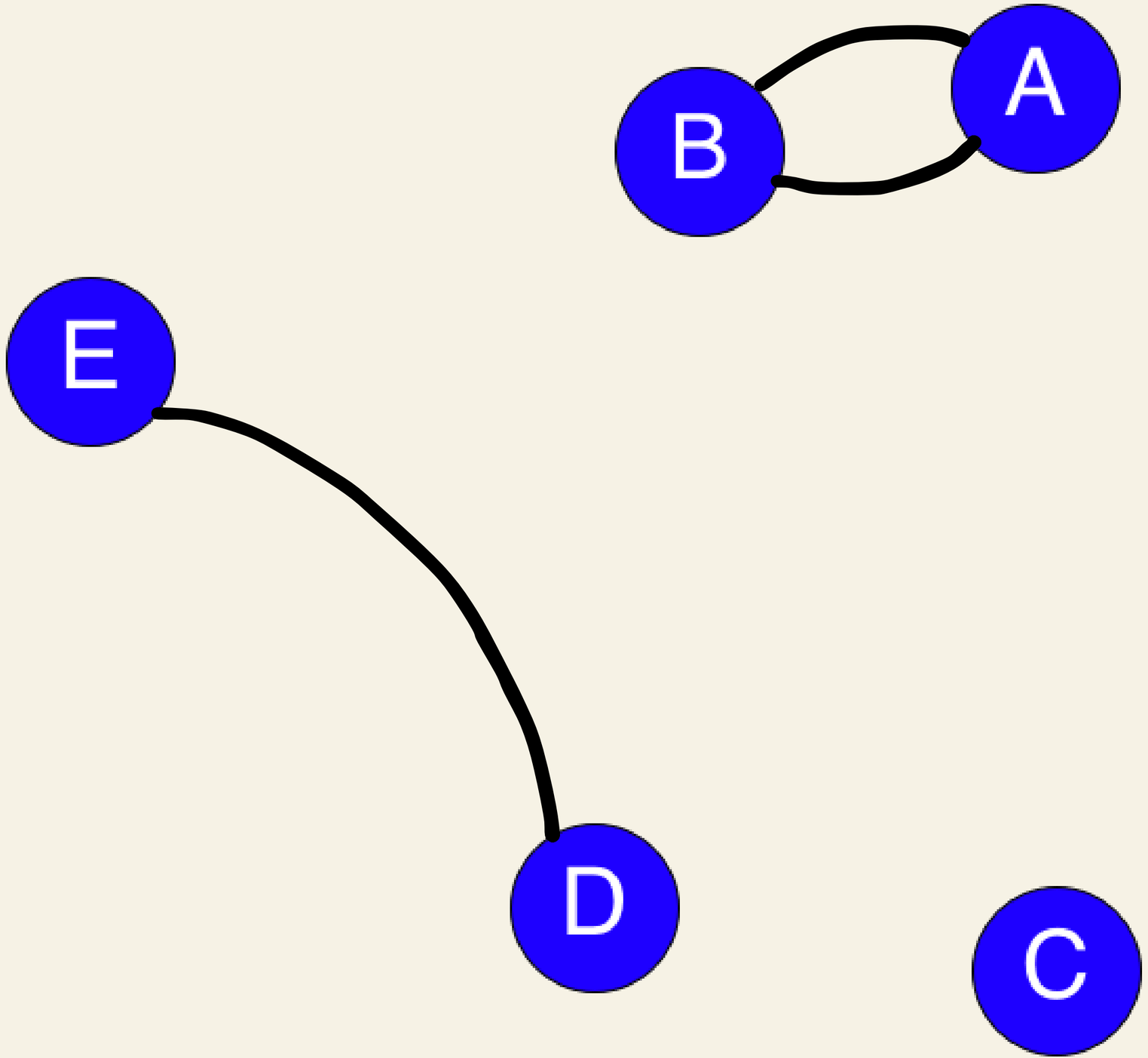
Stack

C
D
E
B
E



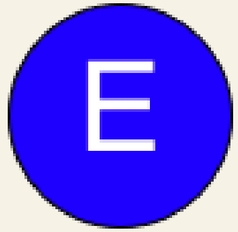
Stack

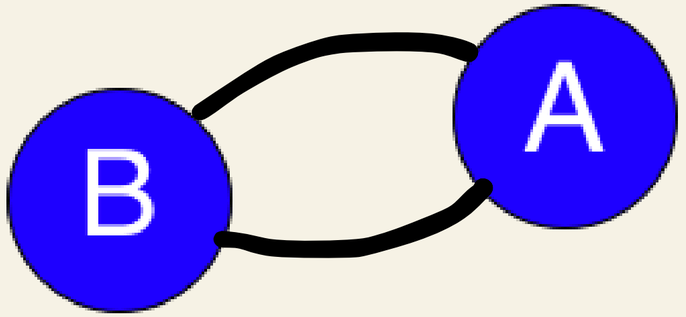
D
C
D
E
B
E

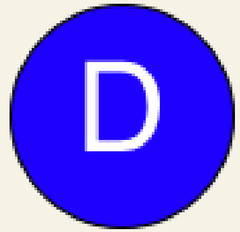


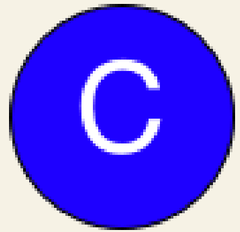
Stack

E
D
C
D
E
B
E









Stack

D

C

D

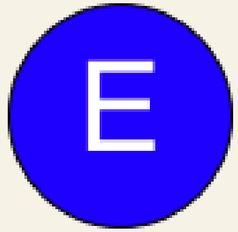
E

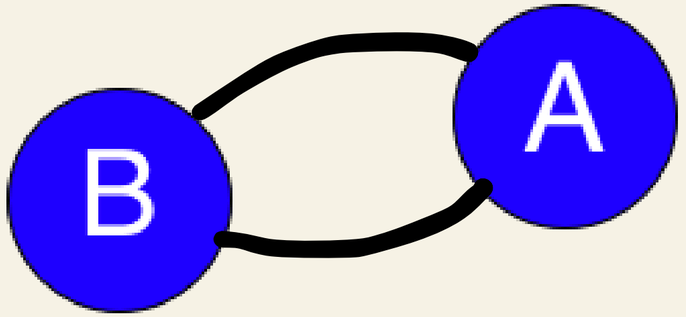
B

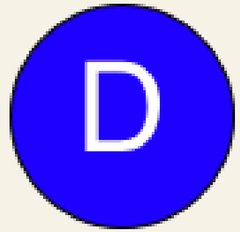
E

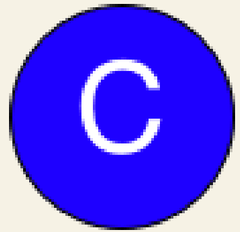
Circuit

E









Stack

C

D

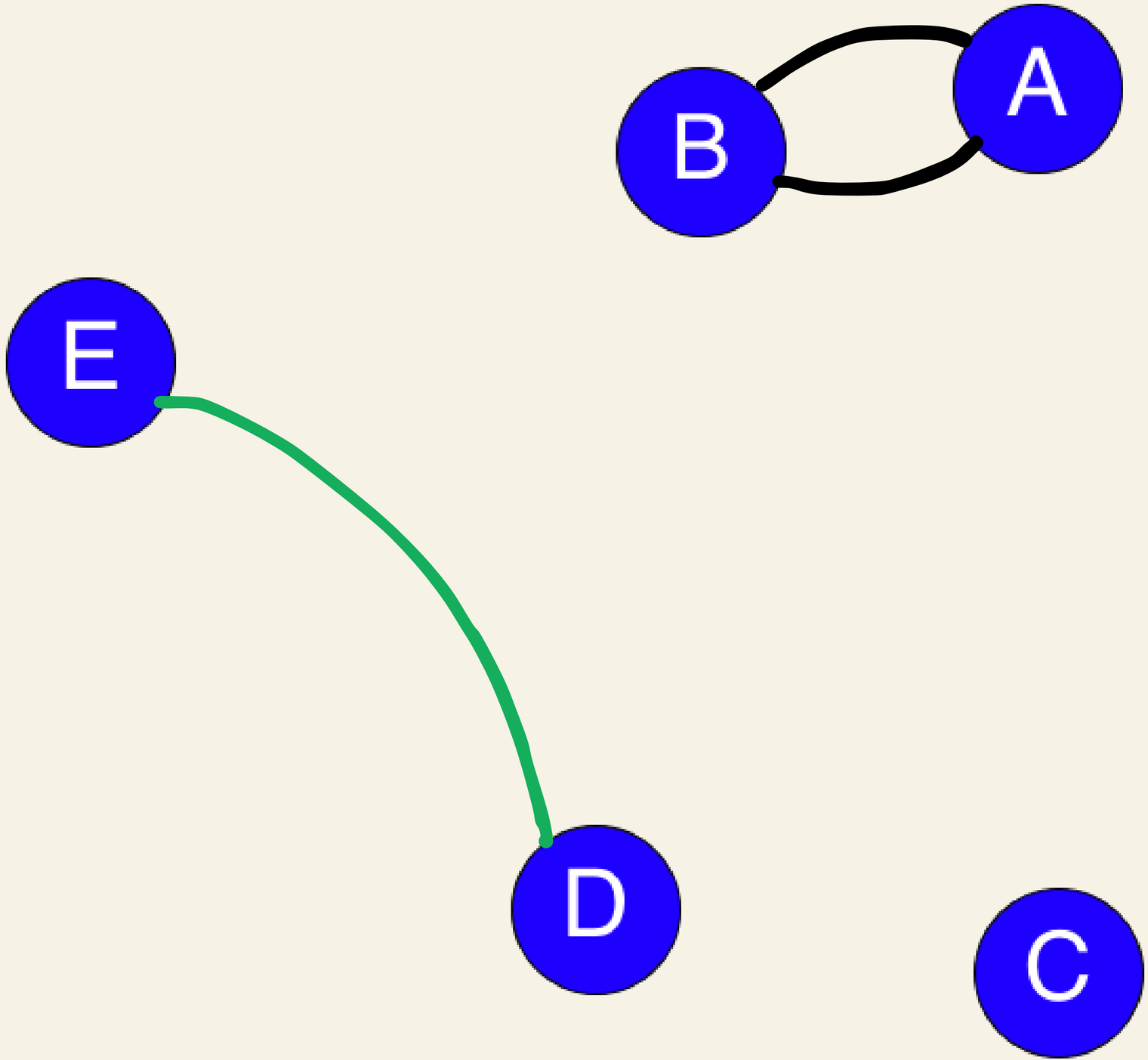
E

B

E

Circuit

E, D



Stack

D

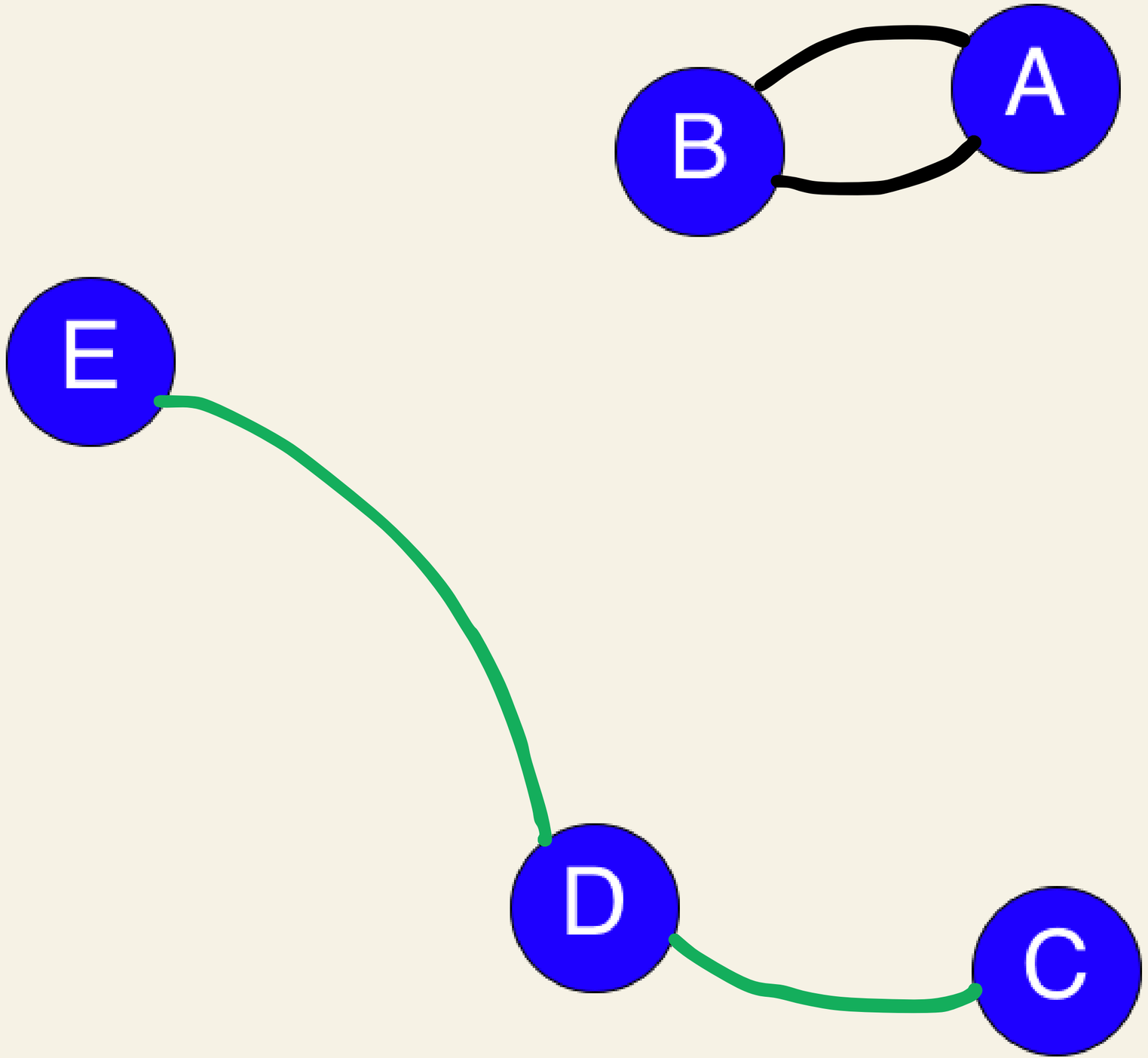
E

B

E

Circuit

E, D, C



Stack

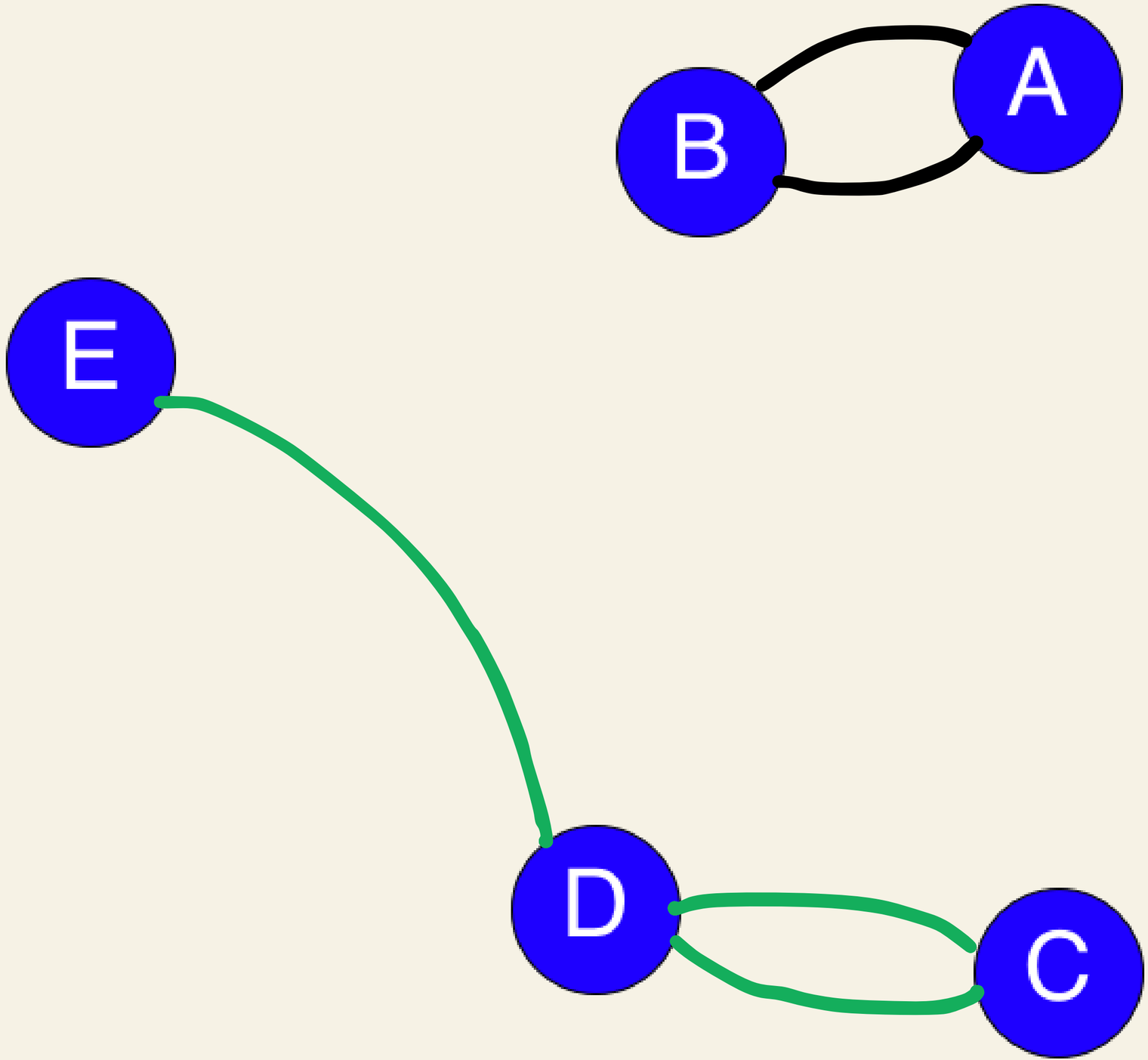
E

B

E

Circuit

E, D, C, D



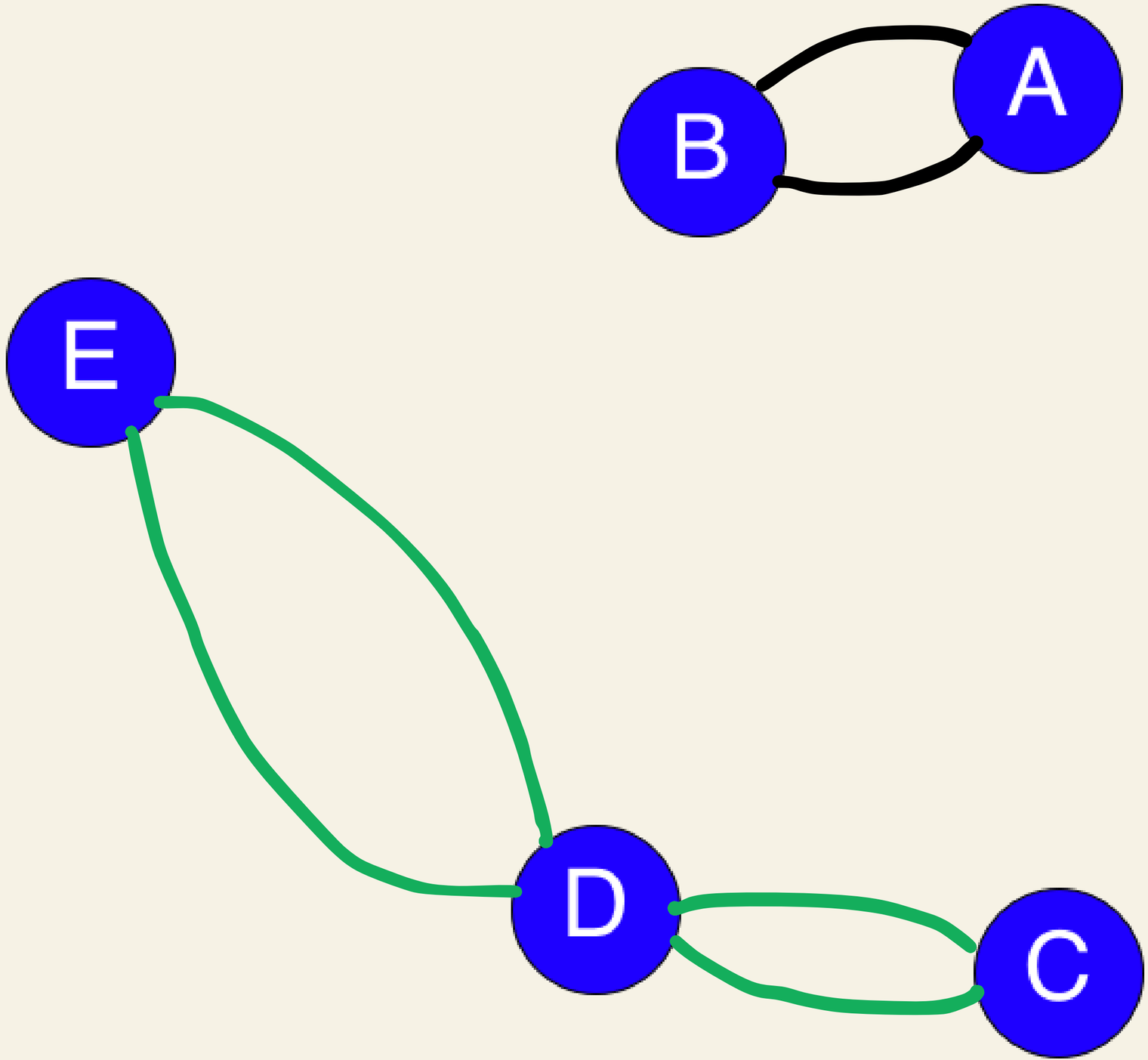
Stack

B

E

Circuit

E, D, C, D, E

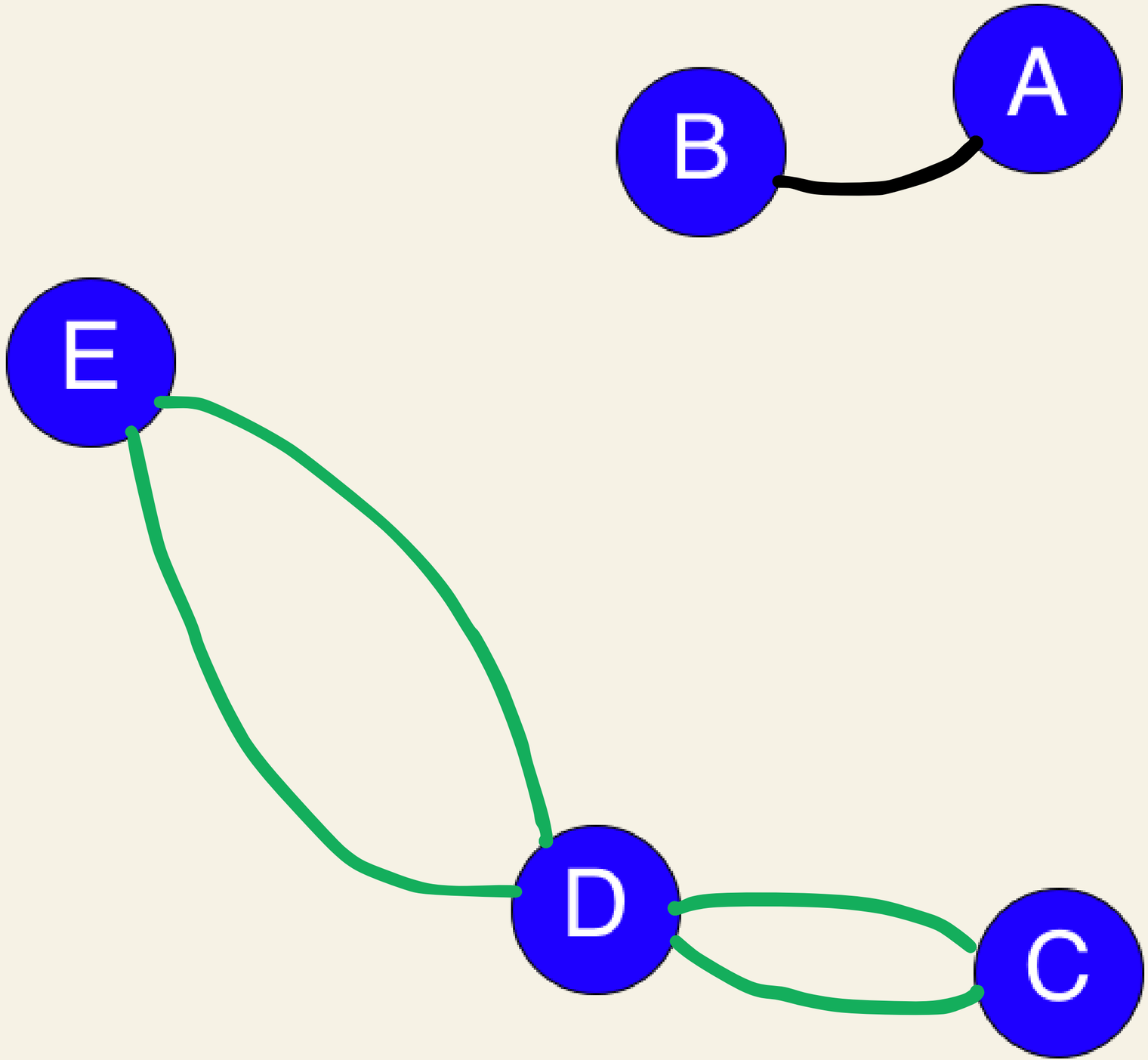


Stack

A
B
E

Circuit

E, D, C, D, E



Stack

B

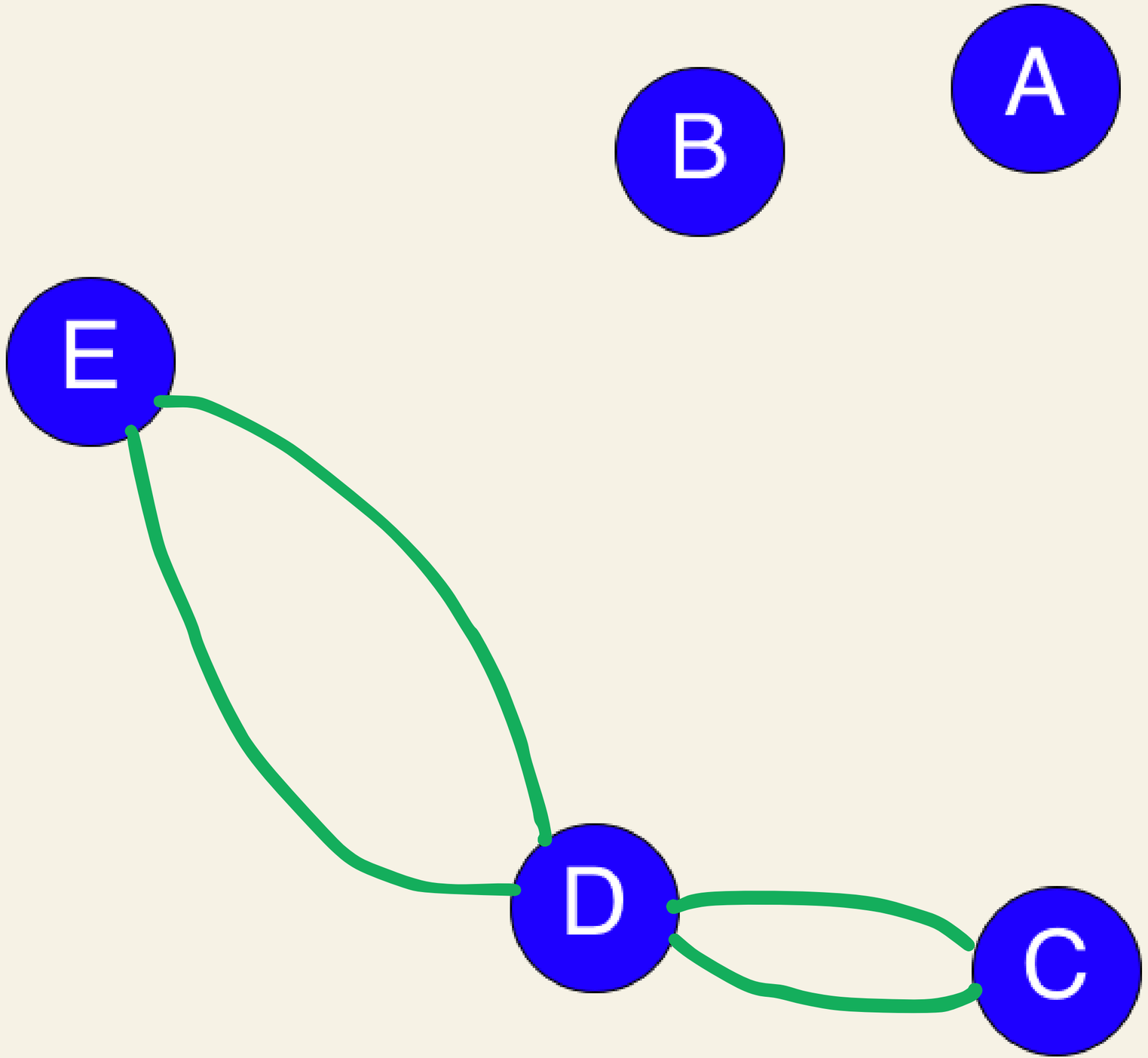
A

B

E

Circuit

E, D, C, D, E

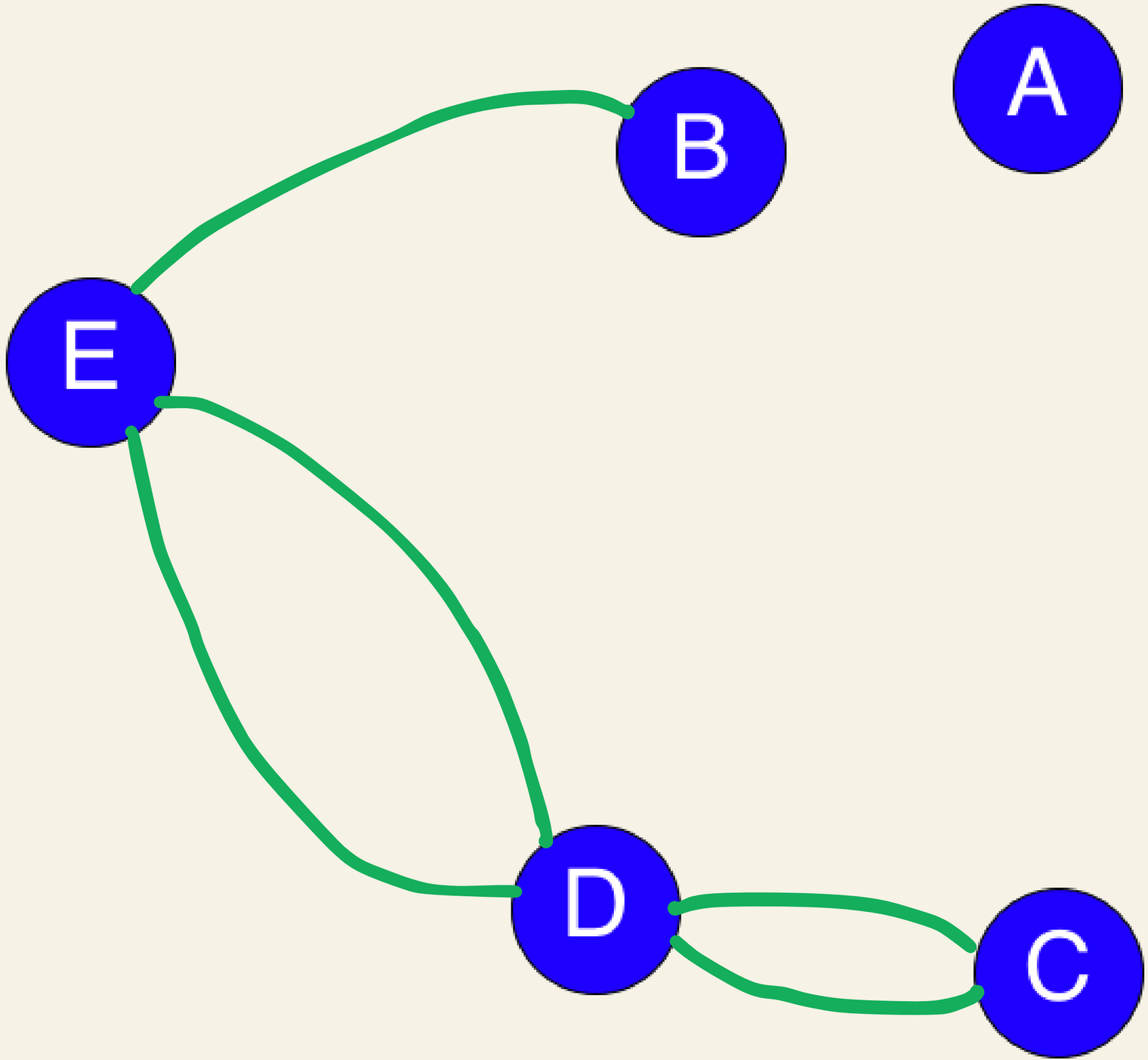


Stack

A
B
E

Circuit

E, D, C, D, E, B



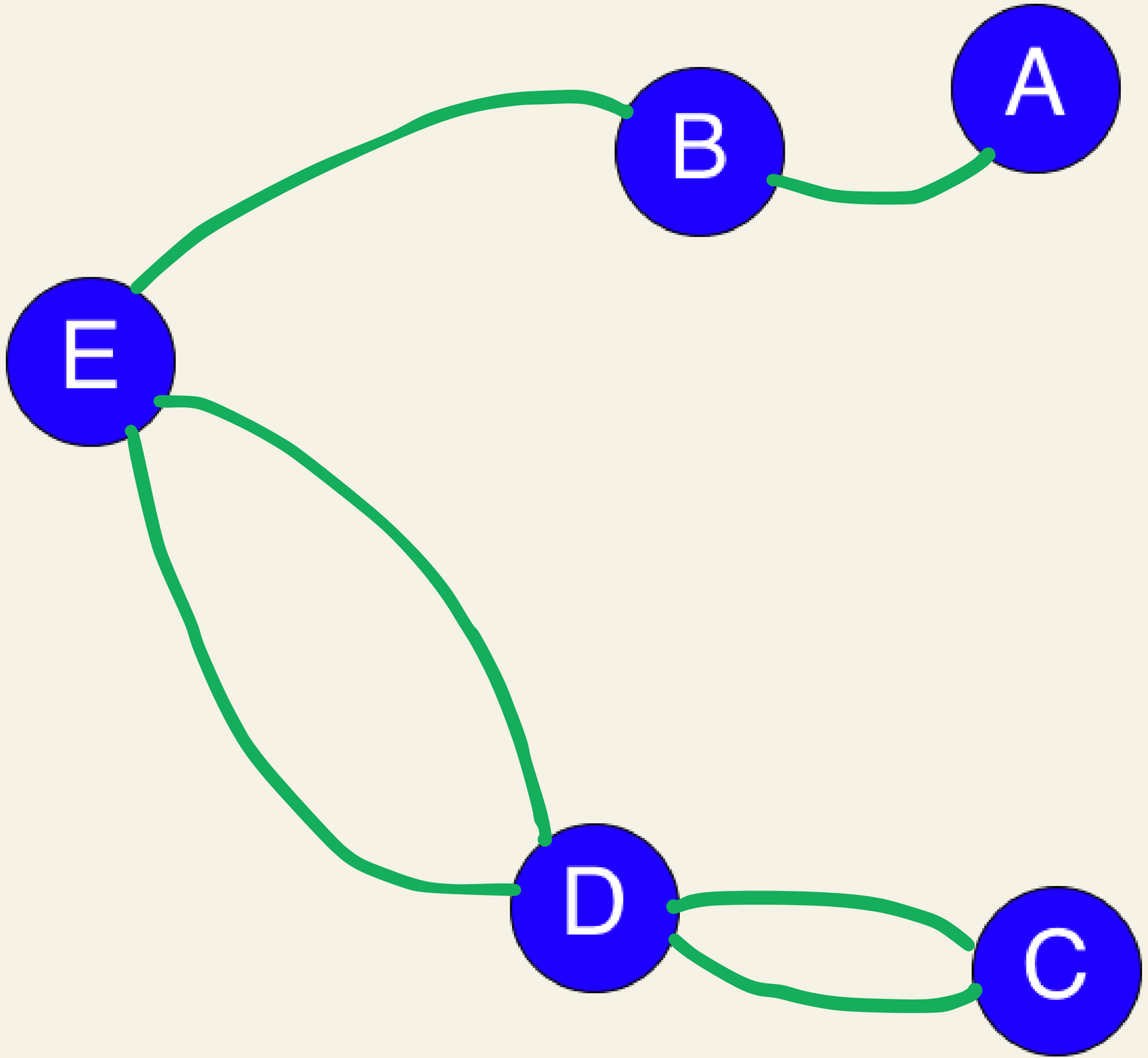
Stack

B

E

Circuit

E, D, C, D, E, B, A

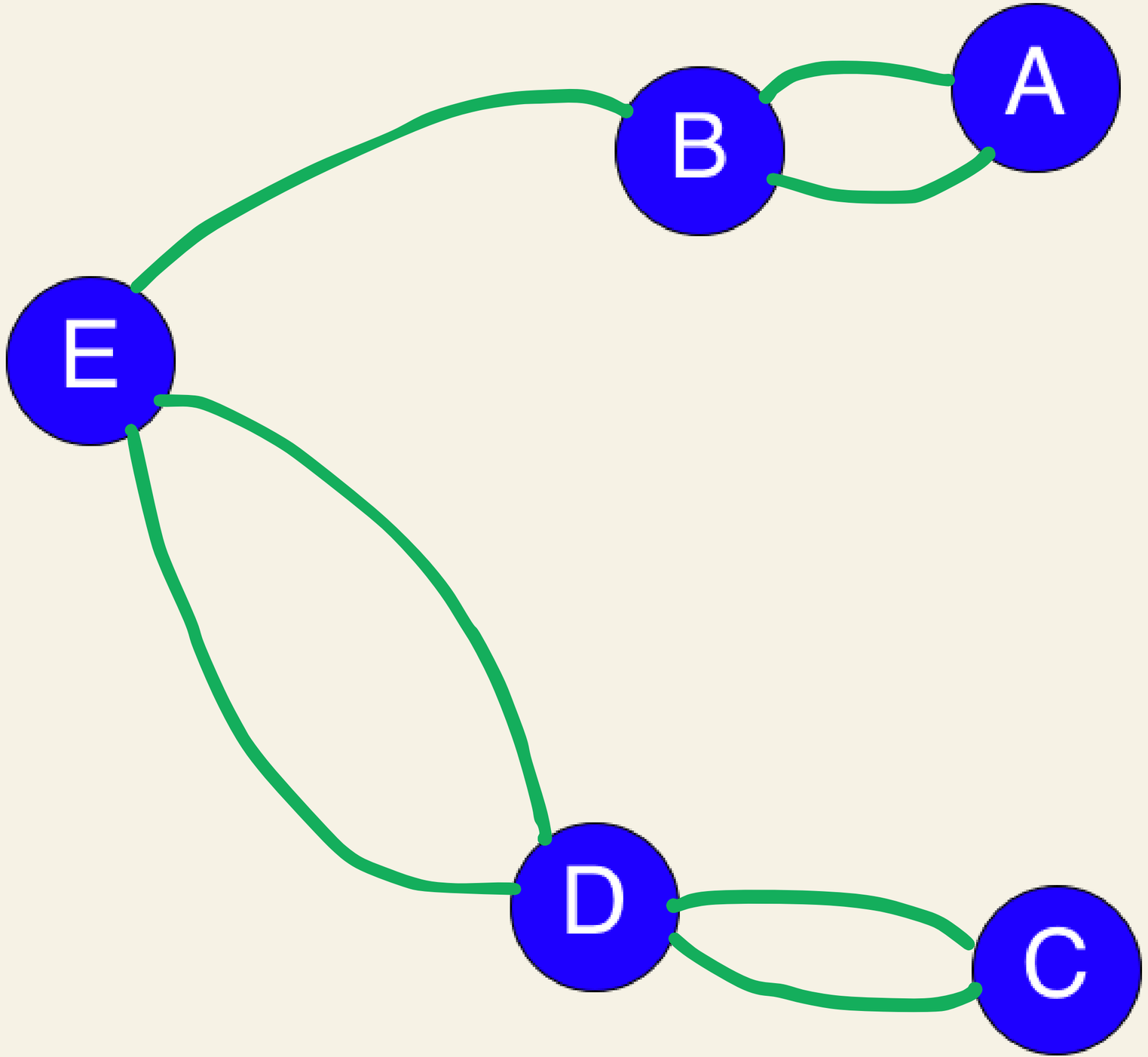


Stack

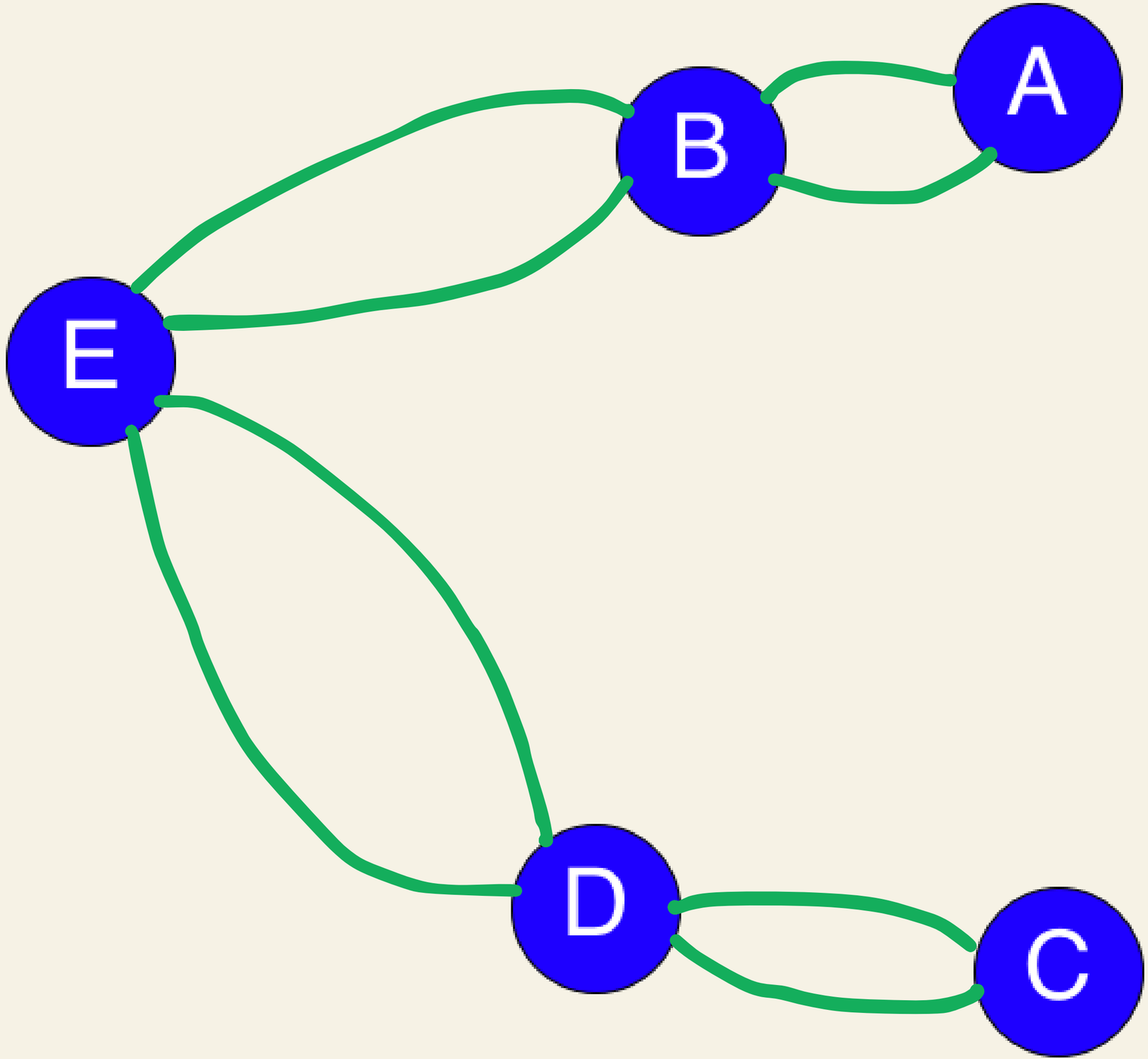
E

Circuit

E, D, C, D, E, B, A, B



Stack



Circuit

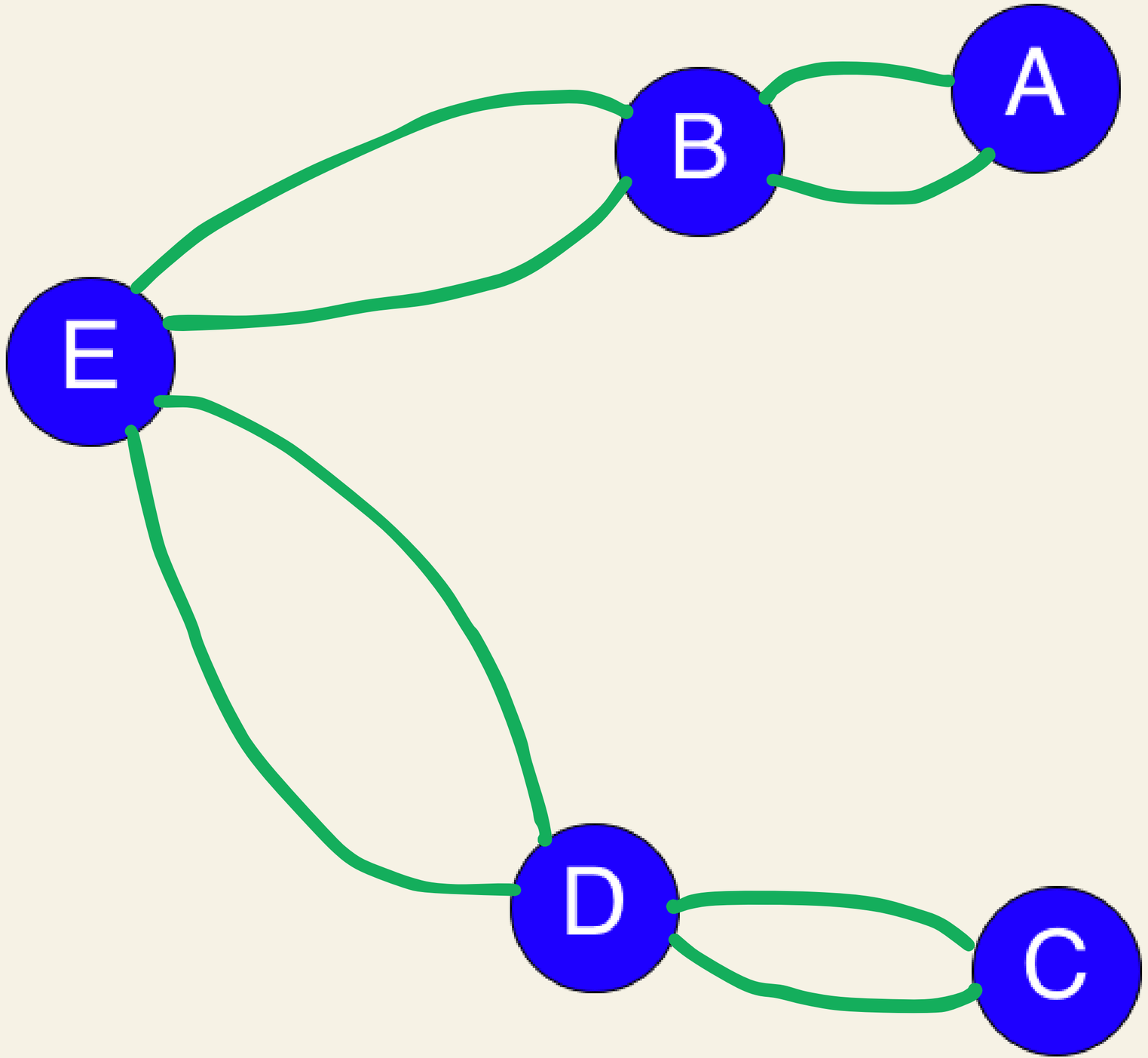
E, D, C, D, E, B, A, B, E

Circuit

E, D, C, D, E, B, A, B, E

Reversed

E, B, A, B, E, D, C, D, E



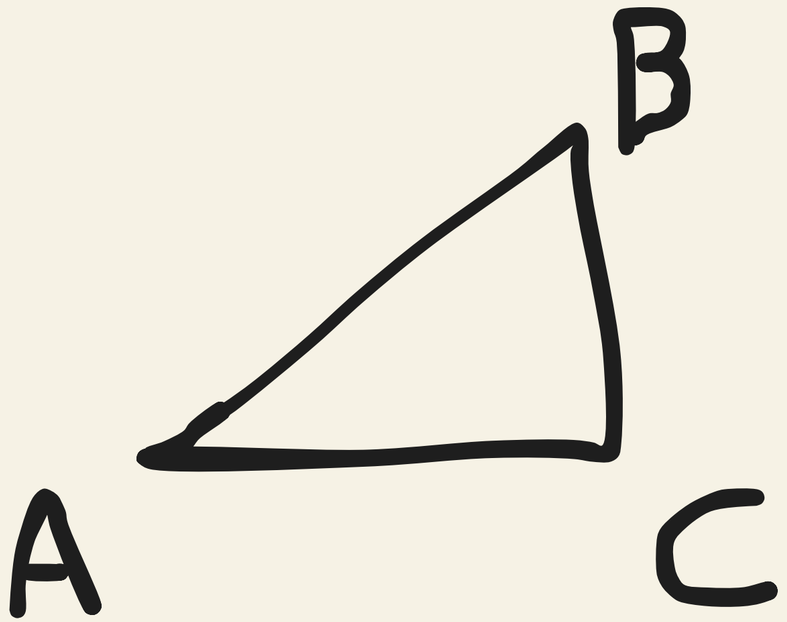
SHORTCUTTING HAMILTONIAN TOUR

Hamiltonian Tour

- A Hamiltonian tour is a cycle that visits every vertex exactly once.

Shortcutting

- Triangle inequality
- Complete graph

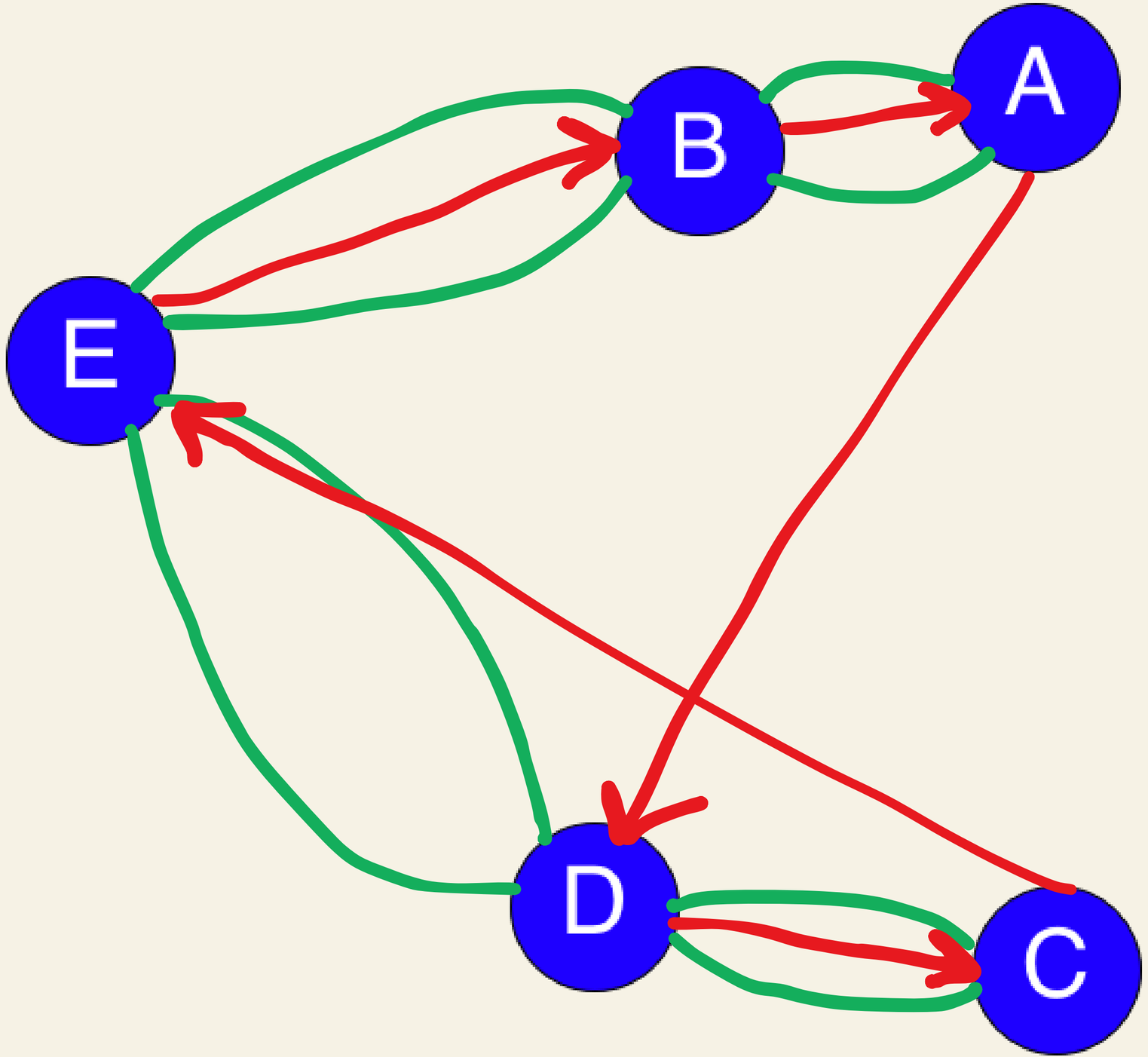


Eulerian Circuit

E, B, A, B, E, D, C, D, E

Hamiltonian Tour

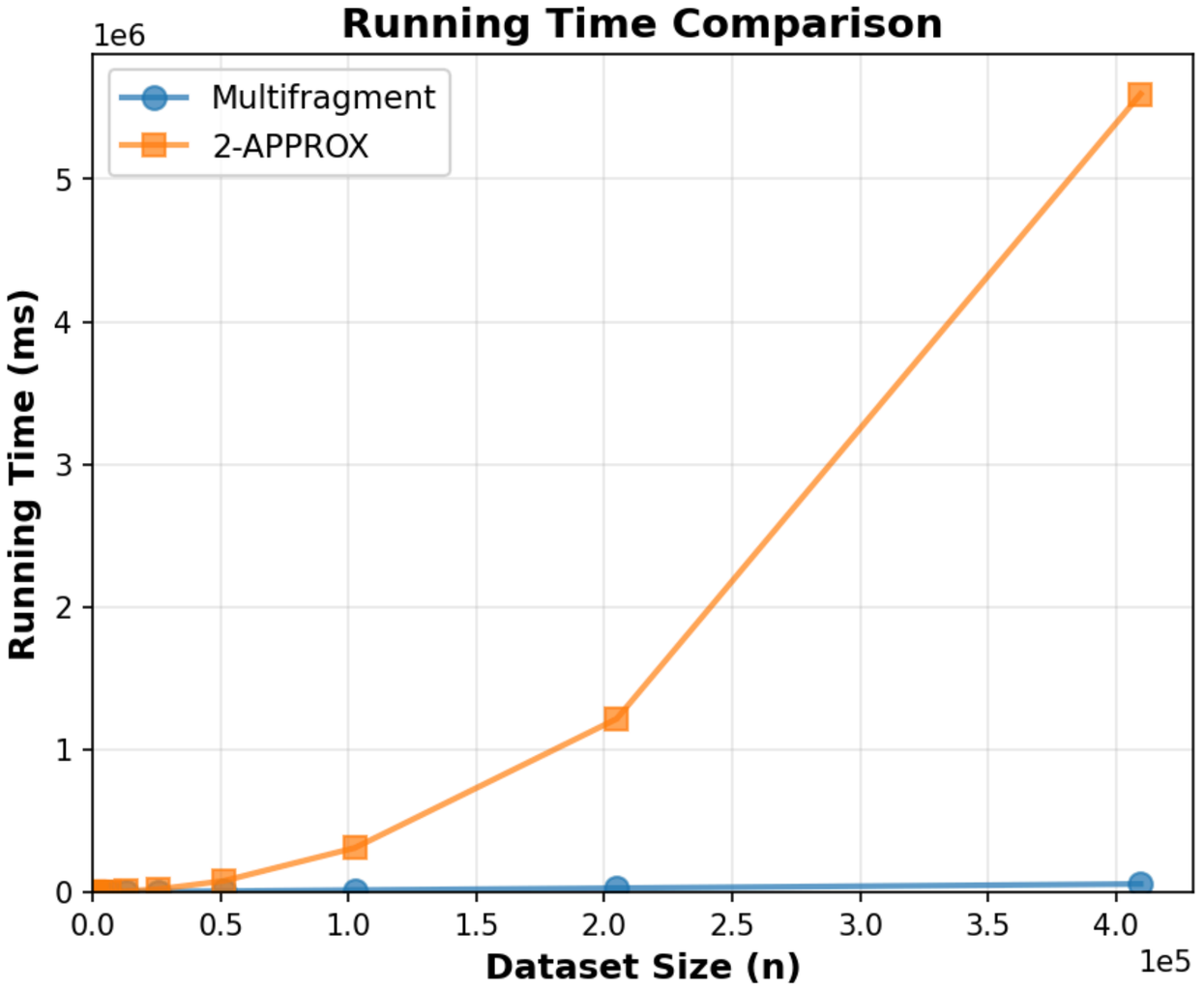
E, B, A, D, C

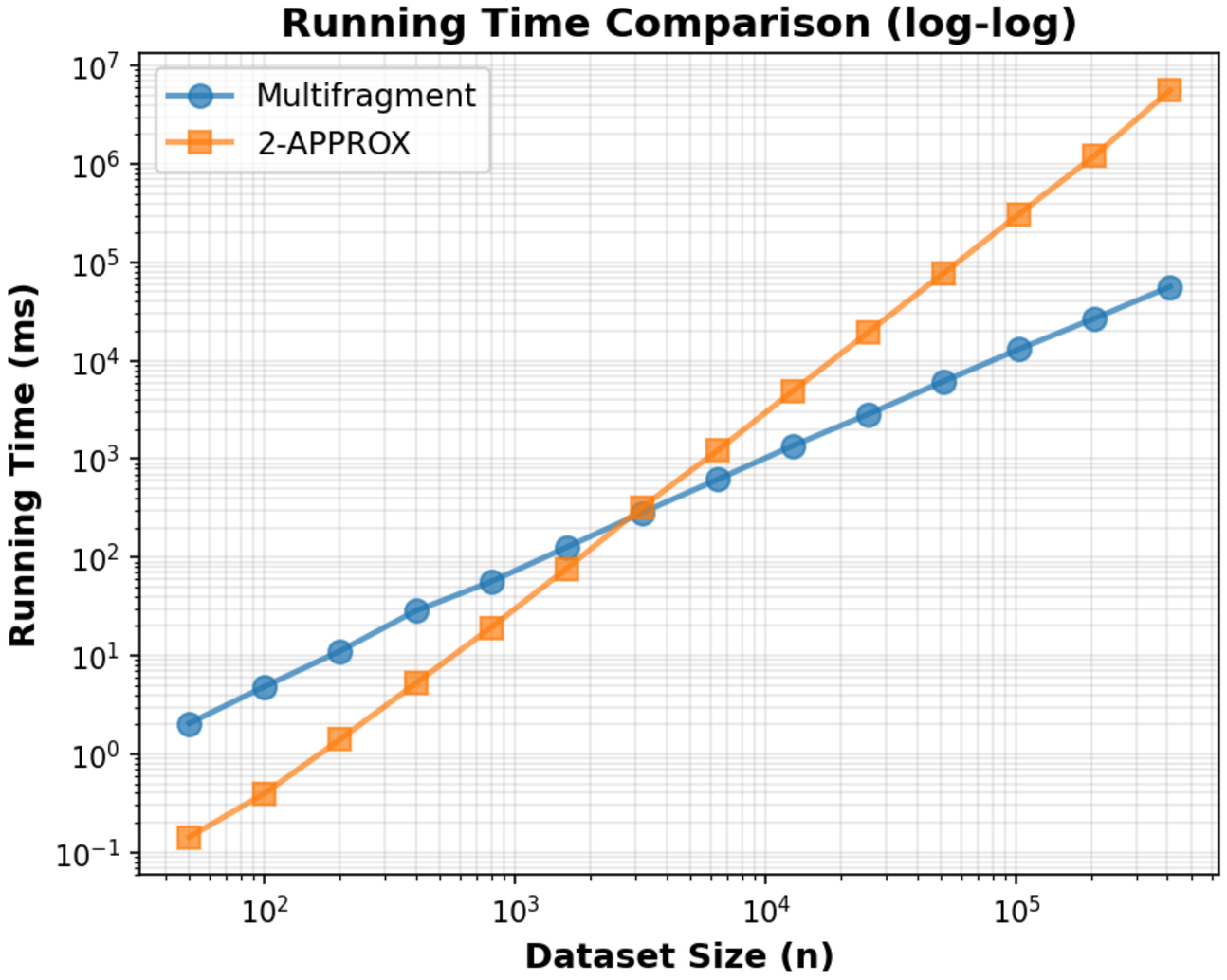


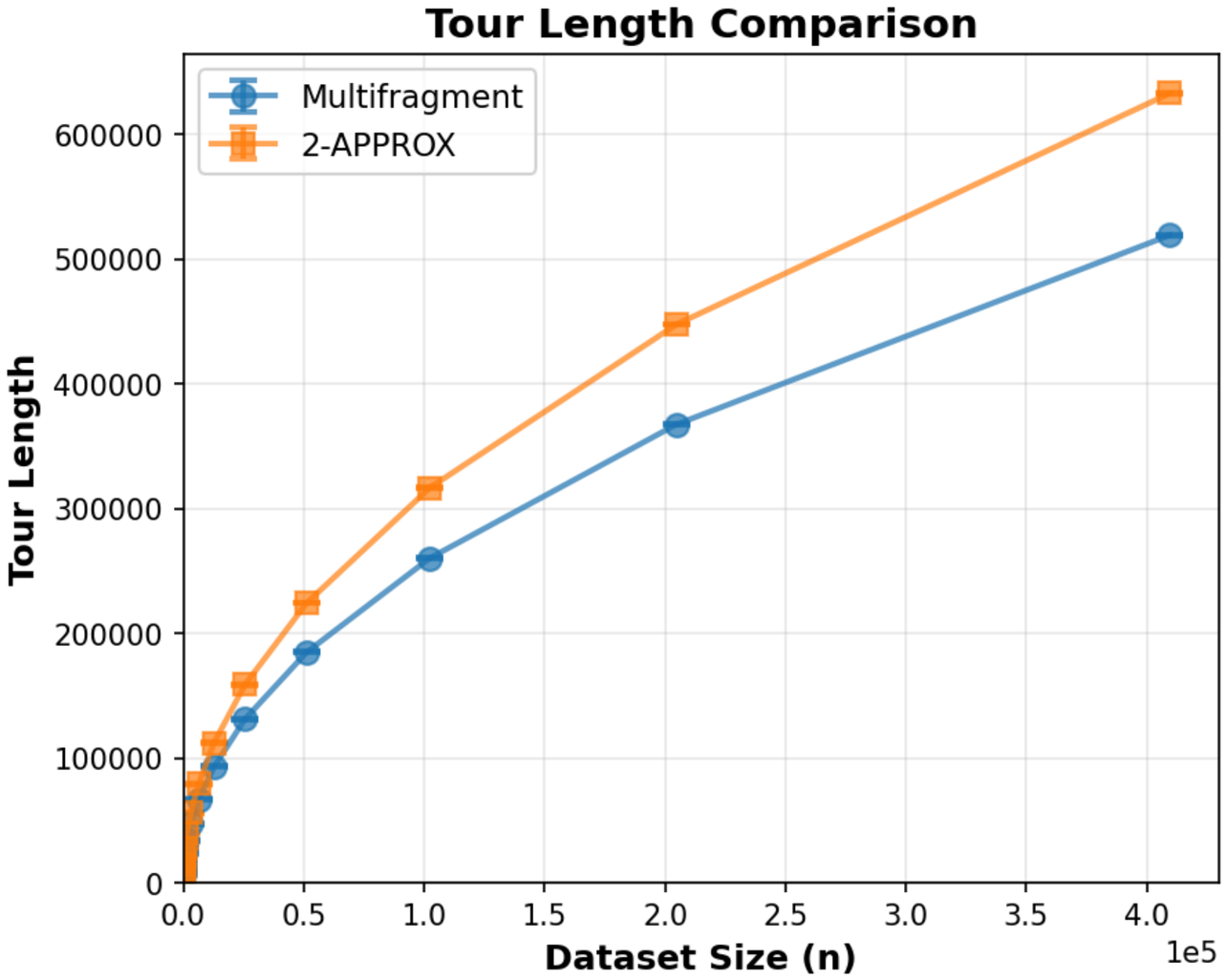
EXPERIMENTS
RANDOM + TSPLIB

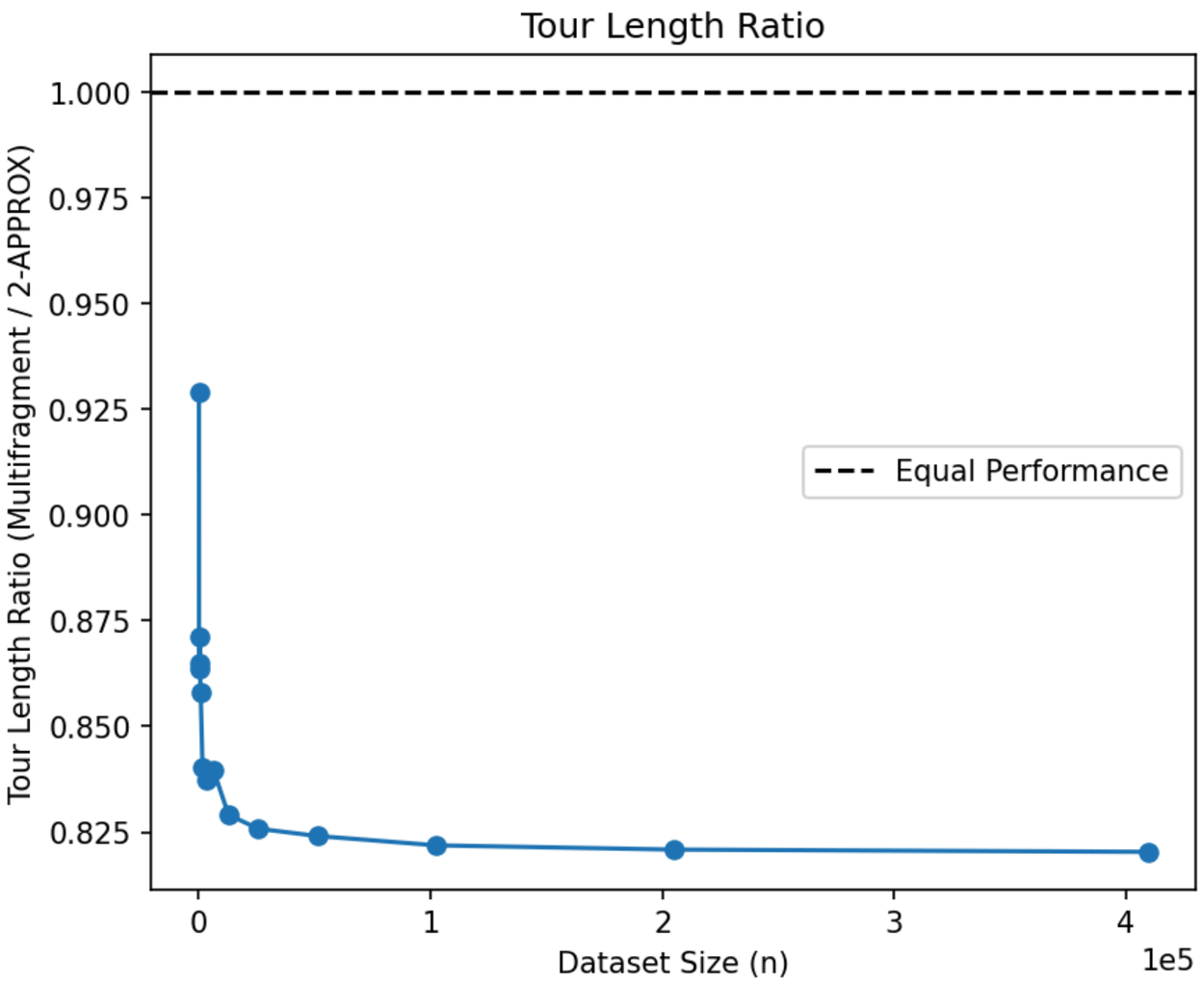
Random Point Sets

- n = 50 continuously doubled up to 409,600
- 10 trials for all n except 409,600
- 5 trials for 409,600


Running Time Comparison
1e6
Multifragment
2-APPROX
Running Time (ms)
0
1
2
3
4
5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
1e5
Dataset Size (n)


Running Time Comparison (log-log)
Multifragment
2-APPROX
Running Time (ms)
10^{7}
10^{6}
10^{5}
10^{4}
10^{3}
10^{2}
10^{1}
10^{0}
10^{-1}
10^{2}
10^{3}
10^{4}
10^{5}
Dataset Size (n)


Tour Length Comparison
Multifragment
2-APPROX
Tour Length
Dataset Size (n)
1e5
0
100000
200000
300000
400000
500000
600000
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

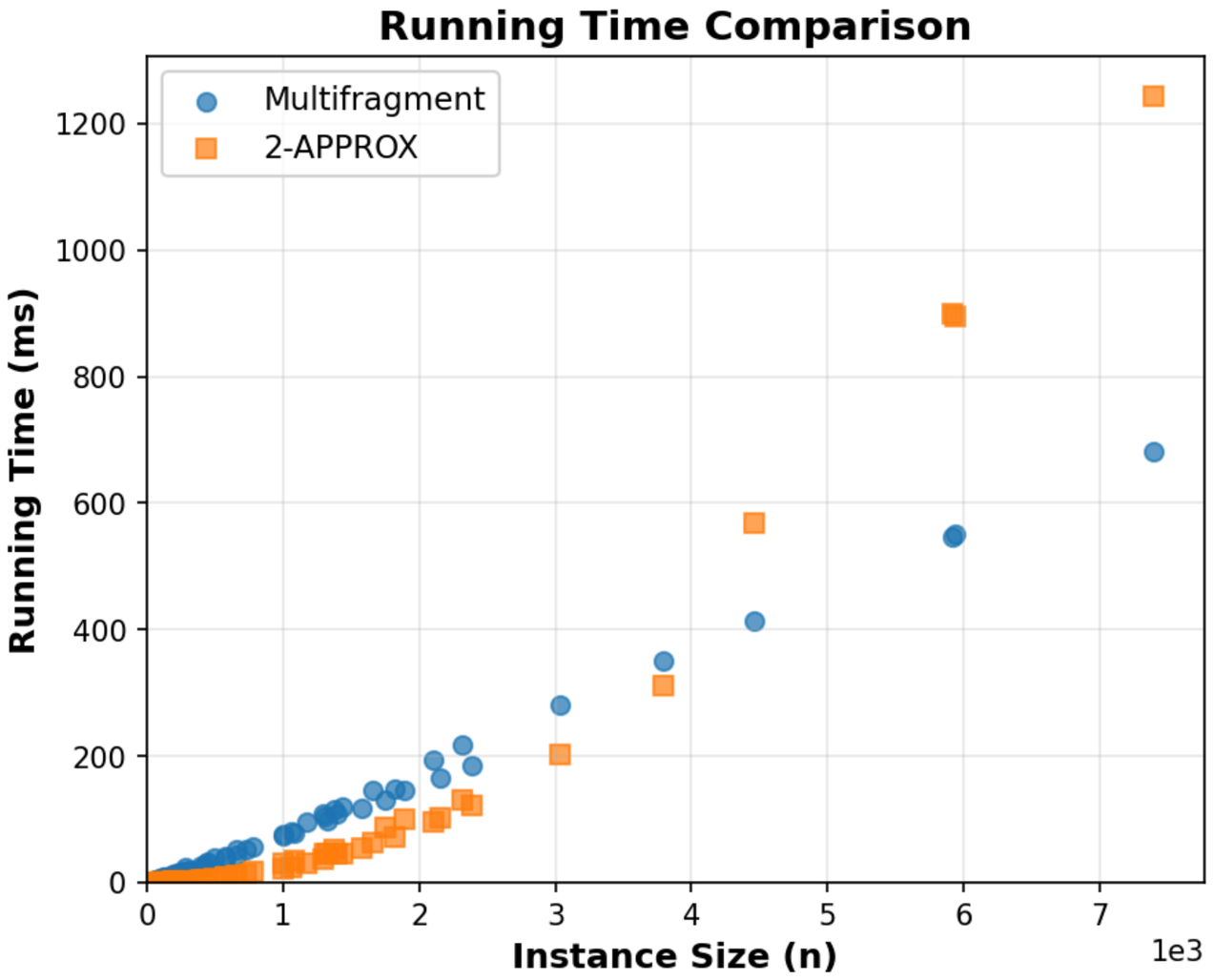

Tour Length Ratio
Tour Length Ratio (Multifragment / 2-APPROX)
Equal Performance
Dataset Size (n)
1e5
1.000
0.975
0.950
0.925
0.900
0.875
0.850
0.825
0
1
2
3
4

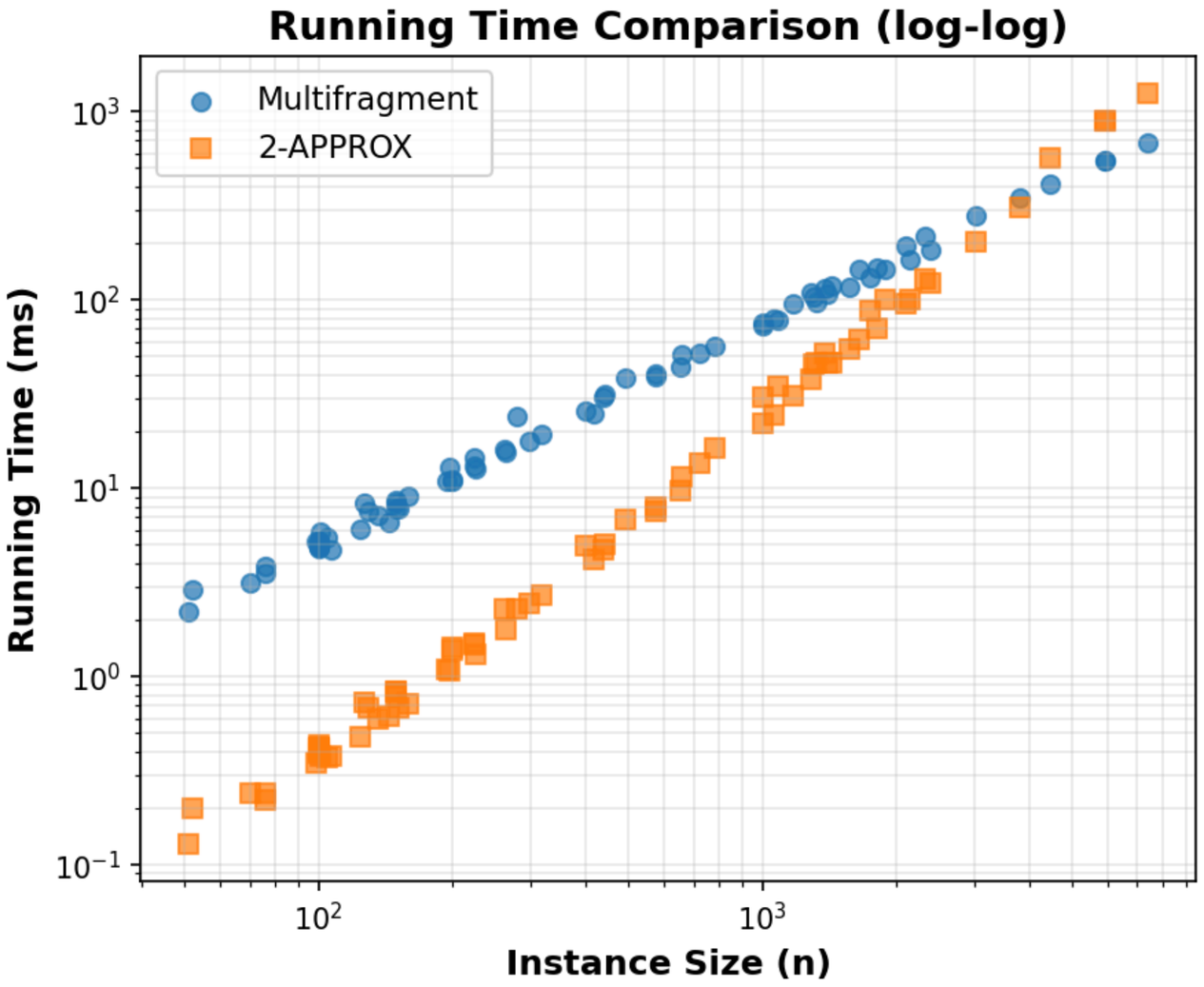
TSPLIB

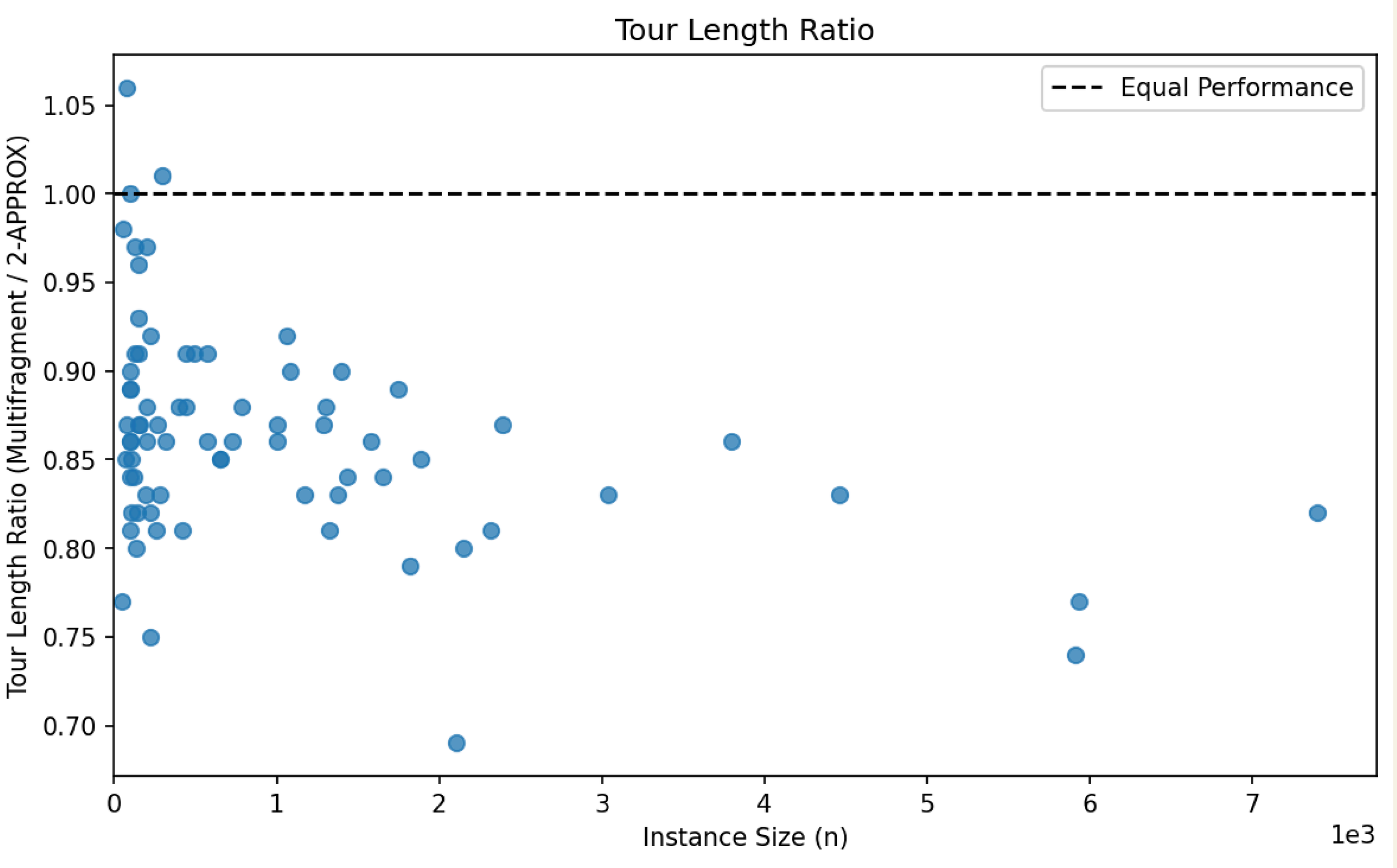
- Various categories
- 73 instances used

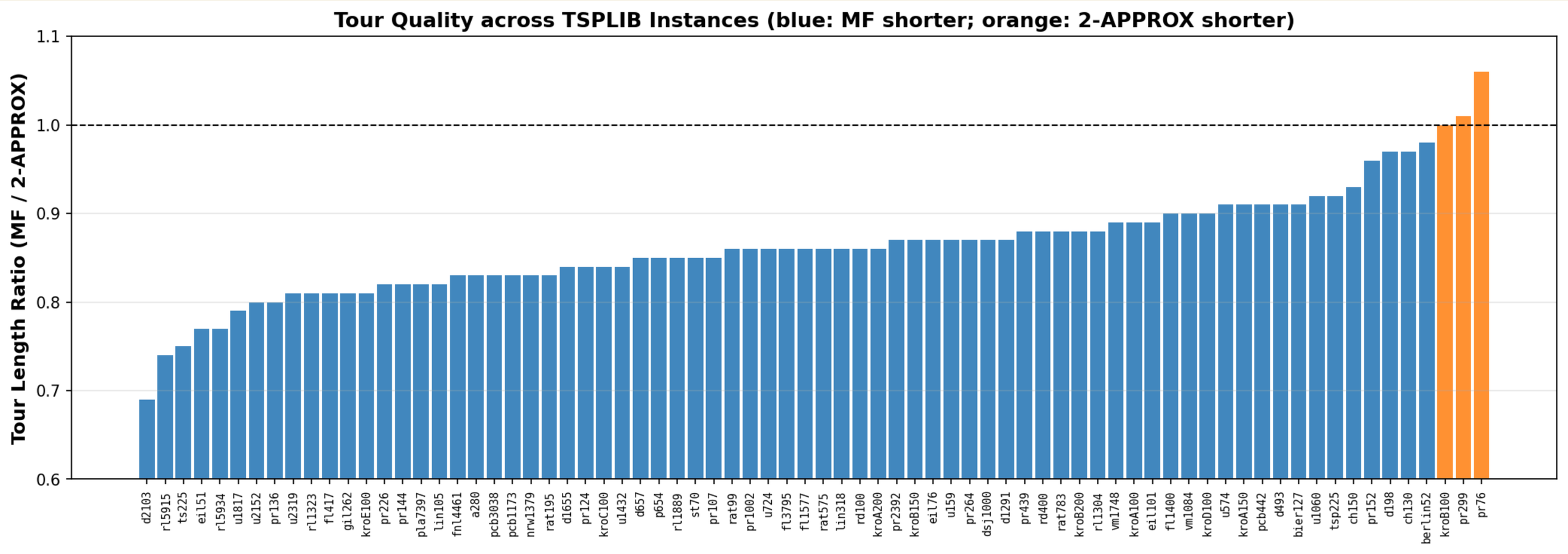
Table 3.1: *Sample TSPLIB Instances*

Category	Size range	Representative instances
Geographic	52–4461	`berlin52`, `nrw1379`, `fnl4461`
PCB drilling	159–3795	`a280`, `d493`, `pcb1173`, `fl1577`
VLSI placement	1084–7397	`vm1084`, `rl1304`, `pla7397`
Padberg/Rinaldi	76–2392	`pr76`, `pr439`, `pr2392`
Rattled grid	99–783	`rat99`, `rat575`, `rat783`
Random / synthetic	51–1000	`eil51`, `kroA100`, `lin318`, `dsj1000`


Running Time Comparison
Multifragment
2-APPROX
Running Time (ms)
Instance Size (n)
1e3


Running Time Comparison (log-log)
Multifragment
2-APPROX
Running Time (ms)
Instance Size (n)


Tour Length Ratio
Equal Performance
Tour Length Ratio (Multifragment / 2-APPROX)
1.05
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0
1
2
3
4
5
6
7
Instance Size (n)
1e3


Tour Quality across TSPLIB Instances (blue: MF shorter; orange: 2-APPROX shorter)
Tour Length Ratio (MF / 2-APPROX)
1.1
1.0
0.9
0.8
0.7
0.6
d2103
rl5915
ts225
ei151
rl5934
u1817
u2152
pr136
u2319
rl1323
fl417
gil262
kroE100
pr226
pr144
pla7397
lin105
fnl4461
a280
pcb3038
pcb1173
nrw1379
rat195
d1655
pr124
kroC100
u1432
d657
p654
rl1889
st70
pr107
rat99
pr1002
u724
fl3795
fl1577
rat575
lin318
rd100
kroA200
pr2392
kroB150
ei176
u159
pr264
dsj1000
d1291
pr439
rd400
rat783
kroB200
rl1304
vm1748
kroA100
eil101
fl1400
vm1084
kroD100
u574
kroA150
pcb442
d493
bier127
u1060
tsp225
ch150
pr152
d198
ch130
berlin52
kroB100
pr299
pr76

Summary of Experiments

- Multifragment outperforms 2-approx in both runtime and in tour length

CONCLUSIONS + FUTURE WORK

Research Question 1

- RQ1: In practice, what are the tradeoffs in approximation performance, runtime, and implementation simplicity between these algorithms?

Research Question 2

- RQ2: In theory, NNC gives a speedup for the MFH. In practice, how much of a speedup is there?

Future Work

- Fully implement Christofides
- More robust experimentation
- Test in higher dimensions

Conclusions

- Worst case theoretical bound doesn’t mean an algorithm performs that way in the real world