Part 1

Q1

(a).

	start10	start12	start20	start30	start40
UCS	2565	Mem	Mem	Mem	Mem
IDS	2407	13812	5297410	Time	Time
A*	33	26	915	Mem	Mem
IDA*	29	21	952	17297	112571

(b).
UCS: Though uniform-cost search was complete and could find optimal solutions, the time and space it cost exponentially grow as the path linearly increased. Therefore, it generated thousands of nodes to find a solution with path length 10 and ran out of memory when trying to find a solution with a longer path.

IDS: Compared to uniform-cost search, iterative deepening search cost similar time and less space to find solutions because it iteratively did depth-first search. Therefore, it generated a similar number of nodes to find a path length 10 solution and run overtime to find solutions with path lengths longer than 30.

A*: A* search is an informed search, so it did not need to generate as many nodes as uninformed searches. However, it kept all expended nodes in memory. It ran out of memory when finding solutions with a path length of more than 30.

IDA*: Iterative deepening A* search is a combination of iterative deepening search and A* search. So, it generated fewer nodes and won't run out of memory when finding solutions with a path length of more than 30.

Q2

	start50		start60		start64	
IDA*	50	14642512	60	321252368	64	1209086782
1.2	52	191438	62	230861	66	431033
1.4	66	116174	82	3673	94	188917
1.6	100	34647	146	55626	162	235852
Greedy	164	5447	166	1617	184	2174

(b).
Previously, line 43 was

F1 is G1 + H1,

I changed it to

F1 is (2 - 1.2) * G1 + 1.2 * H1,

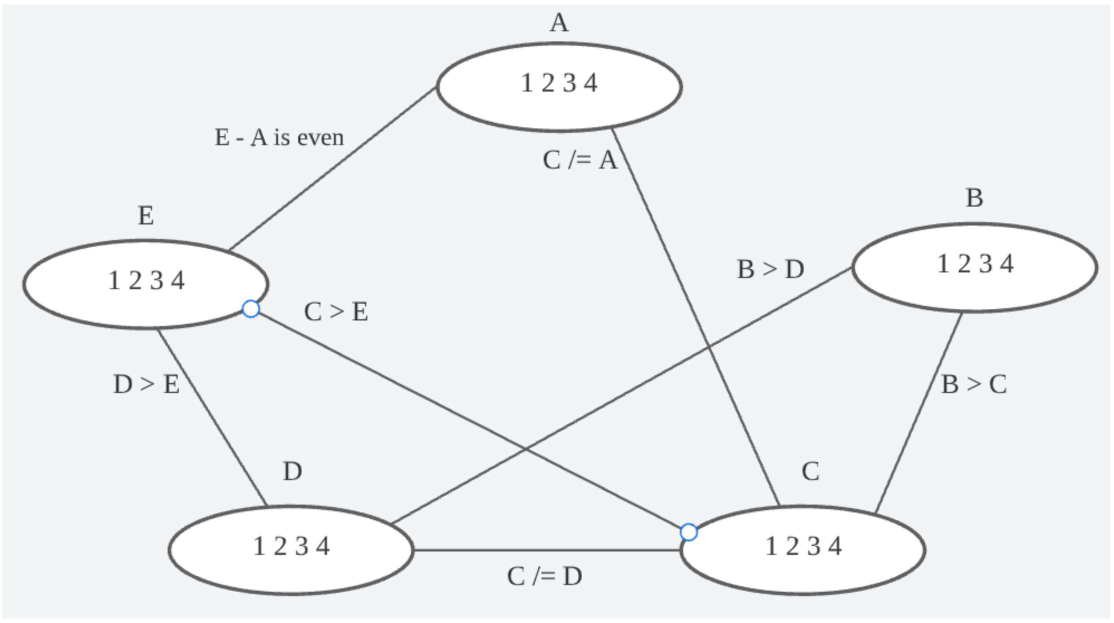
(c).

In general, when we introduced w in IDA*, the algorithm generated fewer nodes and performed much faster. However, greedy is the fastest with generating the fewest nodes. In terms of quality of the solutions, IDA* had best performance that generated solutions with expected steps. The steps of solutions increased as w increased. Greedy algorithms generated solutions with most steps. This is because as w grow, the algorithms put more weight on function H1, which is the cost from current state to the goal, rather than G1, which is the cost from start state to current state. Therefore, greedy method could approach to solutions fast with a lot of cost.

Part 2

Q1

The initial constraint graph is shown as below.



The possible values for nodes are shown as below. In the tables below, elements been labeled with same colors represent the elements that affect each other.

A	B	C	D	E
1 2 3 4	1 2 3 4	1 2 3 4	1 2 3 4	1 2 3 4

Since C > E, C could be at least 2, and E could be at most 3.

A	B	C	D	E	Constraint
1 2 3 4	1 2 3 4	1 2 3 4	1 2 3 4	1 2 3 4	C > E

A	B	C	D	E
1 2 3 4	1 2 3 4	2 3 4	1 2 3 4	1 2 3

Since D > E, D could be at least 2.

A	B	C	D	E	Constraint
1 2 3 4	1 2 3 4	2 3 4	1 2 3 4	1 2 3	D > E

A	B	C	D	E
1 2 3 4	1 2 3 4	2 3 4	2 3 4	1 2 3

Since B > C, B could be at least 3, and C could be as most 3.

A	B	C	D	E	Constraint
1 2 3 4	1 2 3 4	2 3 4	2 3 4	1 2 3	B > C

A	B	C	D	E
1 2 3 4	3 4	2 3	2 3 4	1 2 3

Since B > D, D could be as most 3.

A	B	C	D	E	Constraint
1 2 3 4	3 4	2 3	2 3 4	1 2 3	B > D

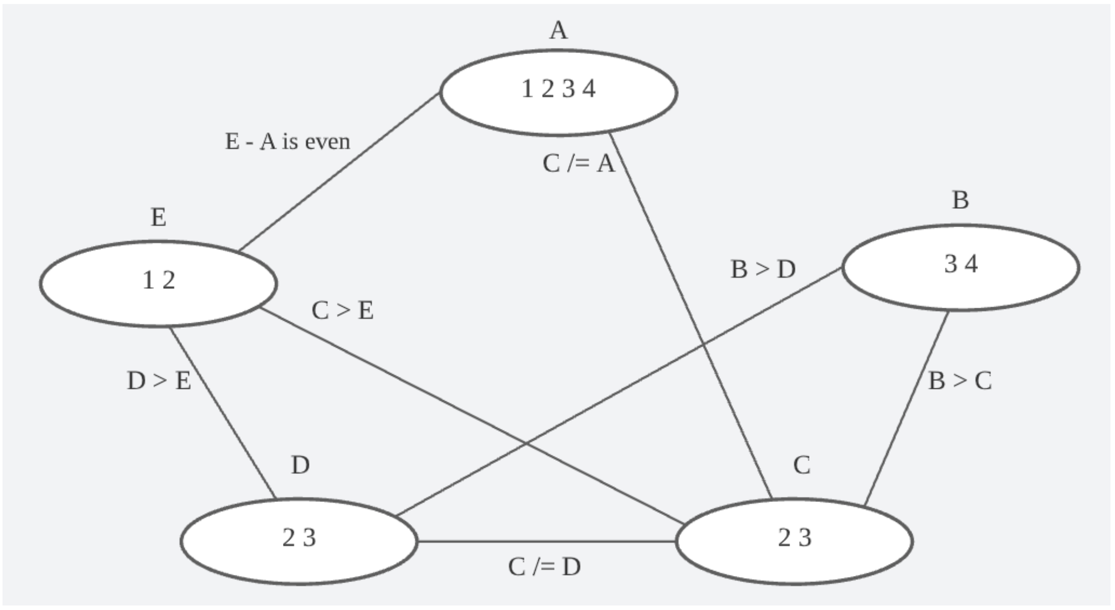
A	B	C	D	E
1 2 3 4	3 4	2 3	2 3	1 2 3

Since C > E, E could be as most 2.

A	B	C	D	E	Constraint
1 2 3 4	3 4	2 3	2 3	1 2 3	C > E

A	B	C	D	E
1 2 3 4	3 4	2 3	2 3	1 2

Now arc consistency stopped. The constraint graph is shown below.



Now we split E’s domain from {1, 2} to {1}, {2}. Assume E is 2. Since C > E, D could be as most 3.

A	B	C	D	E	Constraint
1 2 3 4	3 4	2 3	2 3	2	C > E

A	B	C	D	E
1 2 3 4	3 4	3	2 3	2

Since C /= D, D could only be 2.

A	B	C	D	E	Constraint
1 2 3 4	3 4	2 3	3	2	C /= D

A	B	C	D	E
1 2 3 4	3 4	2	3	2

Since D > E, E could not be 2.

A	B	C	D	E	Constraint
1 2 3 4	3 4	2	3	2	C /= D

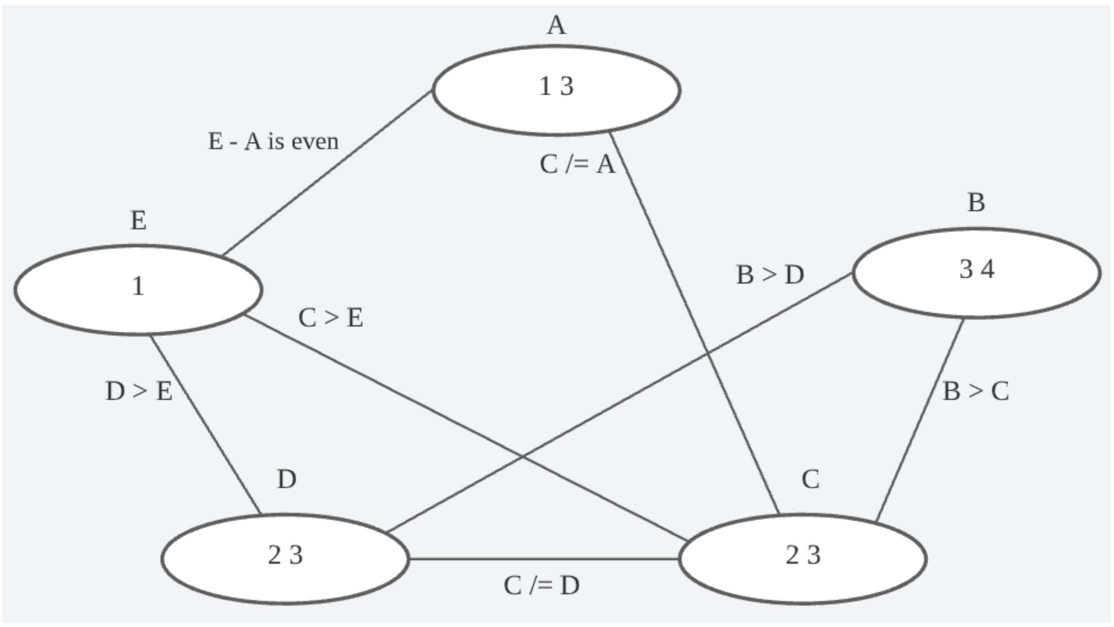
A	B	C	D	E
1 2 3 4	3 4	2	3	

Therefore, no solution found. Thus, E cannot be 2. Then we assume E is 1. Since E – A is even, A could not be 2 and 4.

A	B	C	D	E	Constraint
1 2 3 4	3 4	2 3	2 3	1	E – A is even

A	B	C	D	E
1 3	3 4	3	2 3	1

Now arc consistency stopped. The constraint graph is shown below.



Now we split A's domain from {1, 3} to {1}, {3}. Assume A is 3. Since C /= A, C could not be 3.

A	B	C	D	E	Constraint
3	3 4	2 3	2 3	1	C /= A

A	B	C	D	E
3	3 4	2	2 3	1

Since C /= D, D could not be 2.

A	B	C	D	E	Constraint
3	3 4	2	2 3	1	C /= D

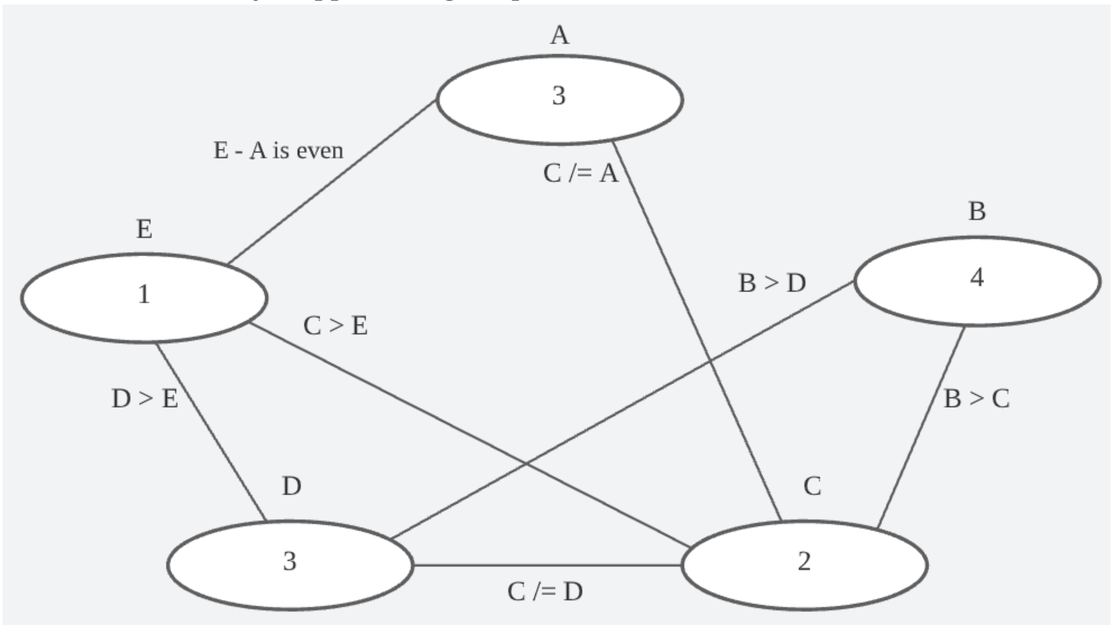
A	B	C	D	E
3	3 4	2	3	1

Since B > D, D could be at least 4.

A	B	C	D	E	Constraint
3	3 4	2	3	1	B > D

A	B	C	D	E
3	4	2	3	1

Now arc consistency stopped. We get a possible solution.



Q2

(a). To eliminate variable A, r1(A, B) and r2(A, C) are removed. r11(B, C) is created on B and C.

(b). To eliminate variable B, r11(B, C), r3(B, D) and r4(B, E) are removed. r12(C, D) and r13(D, E) are created.