Assignment 1

z5271698 Zhaoyan Liu

Part 1 - Search

Question 1: Search Algorithms for the 15-Puzzle

	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: always due to out of memory because it has the largest total number of states expanded, but we cannot know about time running because of the memory out. So it's not efficient.

IDS: It's efficient for memory, after start30 it become timeout.

A*: for small puzzle it's efficient, but for big puzzle it will occupy lots of memory and due to memory out.

IDA*: it is very efficient for time and memory.

Question 2: Heuristic Path Search for 15-Puzzle

	start50		start60		start64	
IDA*	50	14642512	60	321252368	64	1209086782
1.2	G = 52	N =191438	G = 62	N = 230861	G = 66	N = 431033
1.4	G = 66	N = 116342	G = 82	N = 4432	G = 94,	N = 190278
1.6	G = 100	N = 33504	G = 148	N = 55626	G = 162	N = 235848
Greedy	G = 164	N = 5447	G = 166	N = 1617	G = 184	N = 2174

(b) I changed code in line 43:

Code before:

```
42    h(Node1, H1),
43    F1 is G1 + H1,
44    F1 =< F_limit,
```

Code change: when w = 1.2

```
42    h(Node1, H1),
43    F1 is 0.8 * G1 + 1.2 * H1,
44    F1 =< F_limit,
```

,

```
    h(Node1, H1),
    F1 is 0.6 * G1 + 1.4 * H1,
    F1 =< F_limit,
```

w = 1.4 ,

```
42      h(Node1, H1),
43      F1 is 0.4 * G1 + 1.6 * H1,
44      F1 =< F_limit,
```

w =1.6

(d) IDA* has lots of states expanded during the search so that the speed is slow. But length of the path is the shortest one, so the quality of solution is best.

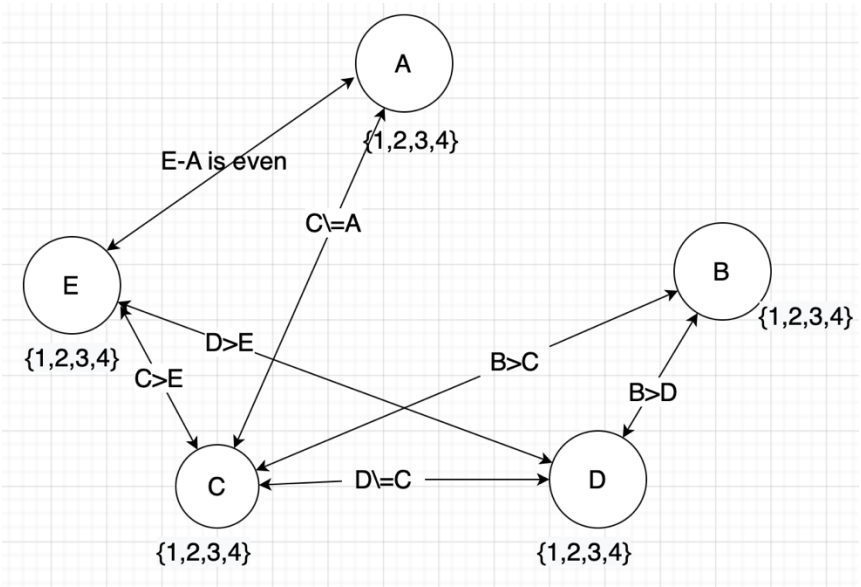
By increasing the w of IDA* the length of the path become large and quality of solution become worse. But we cannot find out what's the connection between w and speed from above table.

For greedy, the speed is the fastest one between this five algorithms because it has least states expanded. But it has the largest length of the path, so the quality of solution is worst.

Part 2 - Constraint Solving

Question 1: Arc Consistency

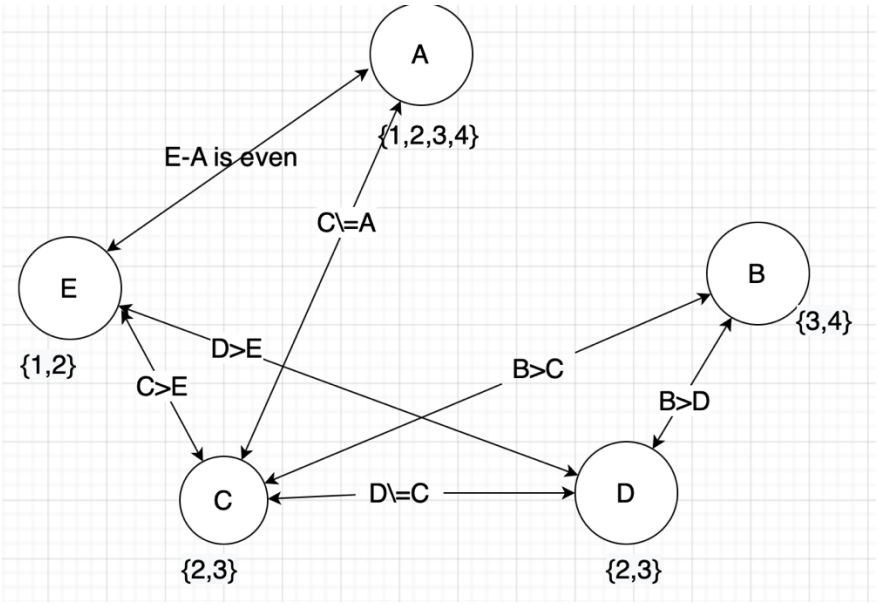
The constraint graph is



- Arc <D,E> remove 1 from domain of D because D>E, D cannot be the smallest number.

- Arc <B,D> remove 1 and 2 from domain of B because B>D.
- Arc <C,E> remove 1 from domain of C because C>E.
- Arc <C,B> remove 4 from domain of C because B>C, C cannot be the largest number.
- Arc <D,B> remove 4 from domain of D because B>D, D cannot be the largest number.
- Arc <E,C> remove 3,4 from domain of E because C>E and C is {2,3}.

the constraint graph after arc consistency has stopped:



Now split a domain. We can split the domain of D. D can be 2 or 3. That is, we **Case 1:** D=2

- Arc <C,D> removes 2 from the domain of C, because C\=D.
- Arc <E,D> removes 2 from the domain of E, because D>E.
- Arc <B,C> removes 3 from the domain of B, because B>C.
- Arc <A,C> removes 3 from the domain of A, because A\=C.
- Arc <A,E> removes 2,4 from the domain of A, because E-A is even.

This results in the solution: **A=1, B=4, C=3, D=2, E=1**

Case 2: D=3

- Arc <C,D> removes 3 from the domain of C, because C\=D.
- Arc <E,C> removes 2 from the domain of E, because C>E.
- Arc <B,D> removes 3 from the domain of B, because B>D.
- Arc <A,C> removes 2 from the domain of A, because A\=C.
- Arc <A,E> removes 4 from the domain of A, because E-A is even.

This results in the solution: **A=1 or 3, B=4, C=2, D=3, E=1**

Question 2: Variable Elimination

(a) To eliminate variable A: $r_1(A, B)$ and $r_2(A, C)$ constraints are removed. A constraint r_{11} is created on B and C ($r_{11}(B, C)$).

(b) To subsequently eliminate B: $r_{11}(B, C)$, $r_4(B, E)$ and $r_3(B, D)$ constraints are removed. A constraint r_{12} is created on D and E ($r_{12}(D, E)$) and constraint r_{13} is created on C and D ($r_{13}(C, D)$).