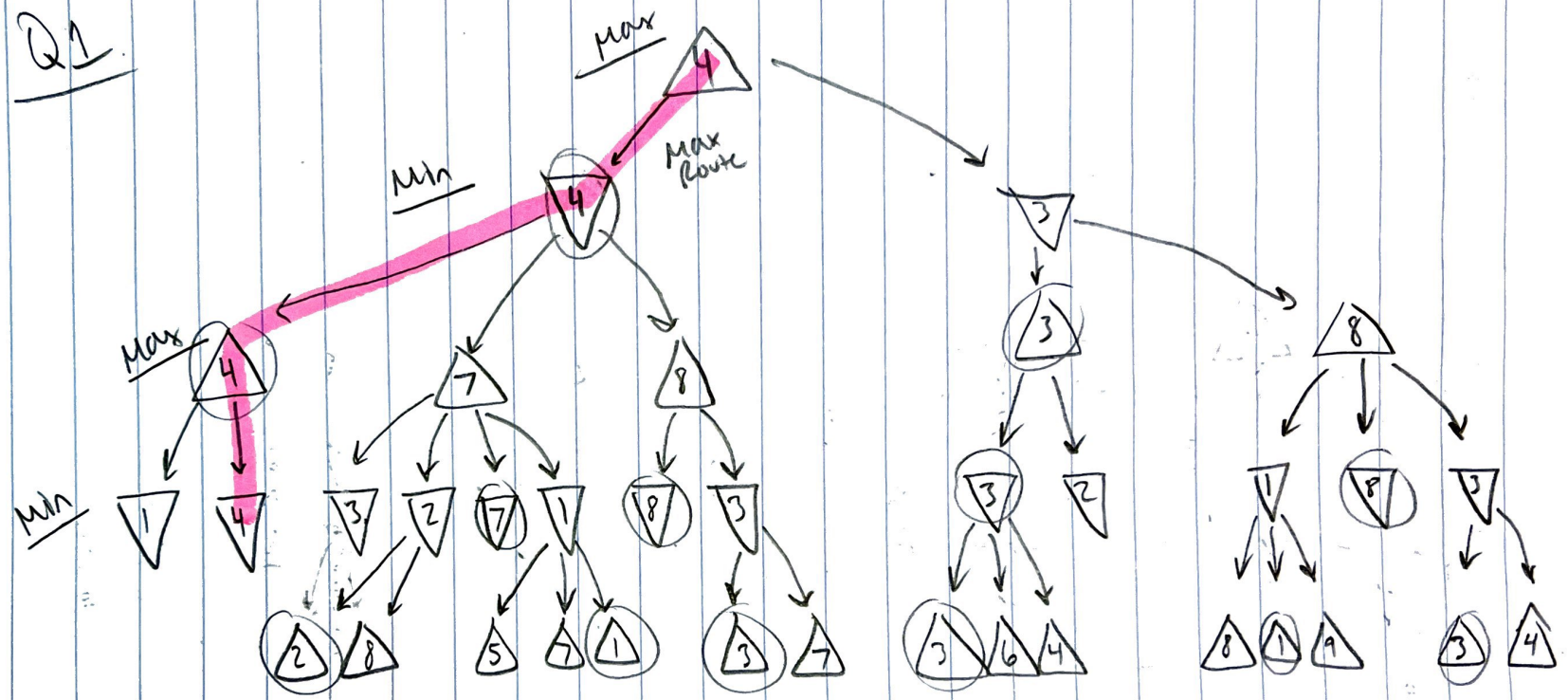
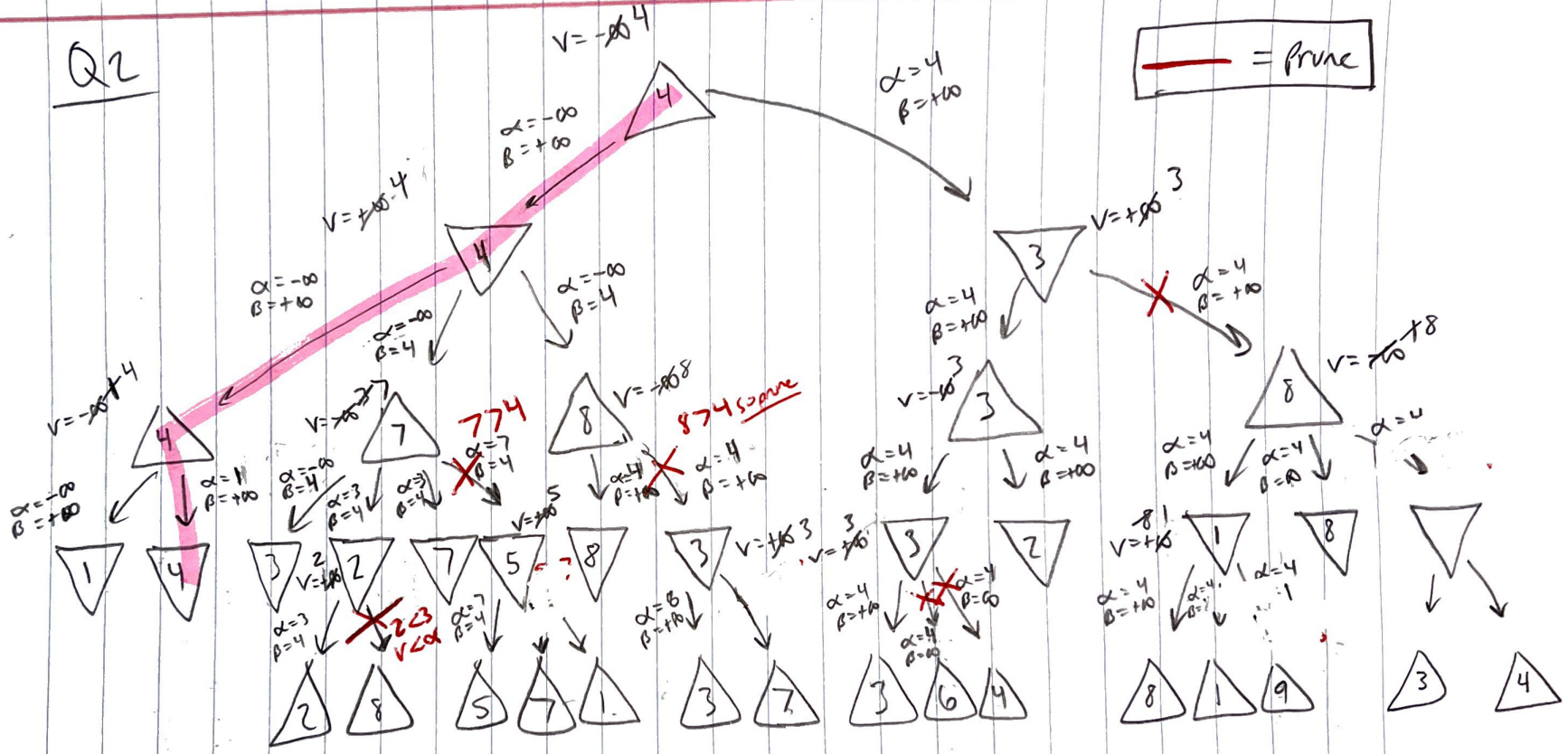
Q1



Q2



Using distributive law

$(P \vee Q) \wedge (P \vee \neg R) \vee (Q \wedge S)$

use distributive on last 2 clauses

$(P \vee Q) \wedge (P \vee \neg R \vee Q) \wedge (P \vee \neg R \vee S)$

c) $(P \rightarrow Q) \rightarrow (R \vee S)$ into DNF

Using $\rightarrow$ definition on first clause

$(\neg P \vee Q) \rightarrow (R \vee S)$

use $\rightarrow$ definition on both clauses

$\neg(\neg P \vee Q) \vee (R \vee S)$

use demorgans on first clause

$(P \wedge \neg Q) \vee (R \vee S)$

Q): Minmax gives an optimal solution despite using DFS because of memory usage. DFS only requires memory of the stack taken, so when we apply a recursive implementation, the execution time remains the same.

If we use BFS, our memory would be taxed and would be way worse than DFS. This is why we use a DFS implementation of a minmax search, and also why it still gives us the optimal solution.

Hw2

$F \to G = \neg F \vee G$

$\neg F \quad \vee \quad G$

Q4.

P	Q	R	$(P \vee R)$	$(Q \vee \neg R)$	$(P \vee R) \wedge (Q \vee \neg R)$	$(P \vee Q)$	$((P \vee R) \wedge (Q \vee \neg R)) \to (P \vee Q)$
T	T	T	T	T	T	T	T
T	T	F	T	T	T	T	T
T	F	T	T	F	F	T	T
T	F	F	T	T	T	T	T
F	T	T	T	T	T	T	T
F	T	F	F	T	F	T	T
F	F	T	T	F	F	F	T
F	F	F	F	T	F	F	T

All true, therefore the conclusion is also true

Q5:

a) $P \vee (Q \wedge \neg R)$ into CNF

Using distributive Law $= (P \vee Q) \wedge (P \vee \neg R)$

b) $P \vee (Q \wedge \neg R) \vee (Q \wedge S)$ into CNF

using distributive law on first 2 to get

$(P \vee Q) \wedge (P \vee \neg R) \vee (Q \wedge S)$

use distributive on last 2 clauses

$(P \vee Q) \wedge (P \vee \neg R \vee Q) \wedge (P \vee \neg R \vee S)$

c) $(P \rightarrow Q) \rightarrow (R \vee S)$ into DNF

Using $\rightarrow$ definition on first clause

$(\neg P \vee Q) \rightarrow (R \vee S)$

use $\rightarrow$ definition on both clauses

$\neg(\neg P \vee Q) \vee (R \vee S)$

use demorgans on first clause

$(P \wedge \neg Q) \vee (R \vee S)$

Q7: Minmax gives an optimal solution despite using DFS because of memory usage. DFS only requires memory of the stack traces so when we apply a recursive implementation, the execution time remains the same.

If we use BFS, our memory would be taxed and would be way worse than DFS. This is why we use a DFS implementation of a minmax search, and also why it still gives us the optimal solution.

6. $R \to (Q \vee S) \Rightarrow \overbrace{\neg R \vee (Q \vee S)}^{C_1}$

$(U \to V) \wedge \neg V \Rightarrow (\neg U \vee V) \wedge \neg V \Rightarrow \neg V \wedge (\neg U \vee V) \Rightarrow (\neg V \wedge \neg U) \vee (\neg V \wedge V)$

$\Rightarrow \underbrace{\neg V}_{C_2} \wedge \underbrace{\neg U}_{C_3}$

$S \to (P \vee V) \Rightarrow \underbrace{\neg S \vee (P \vee V)}_{C_4}$

$(U \vee R) = C_5$

$\neg (P \vee Q) \Rightarrow \underbrace{\neg P}_{C_6} \wedge \underbrace{\neg Q}_{C_7}$

$C_1: \neg R \vee Q \vee S$

$C_2: \neg V$

$C_3: \neg U$

$C_4: \neg S \vee P \vee V$

$C_5: U \vee R$

$C_6: \neg P$

$C_7: \neg Q$

7. Resolution

used

(1,7)	$\neg R \vee S$	(8)	~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~ ~~7~~ ~~8~~ 9 10 11 12
(2,4)	$\neg S \vee P$	(9)	
(6,9)	$\neg S$	(10)	
(3,5)	R	(11)	
(8,10)	$\neg R$	(12)	
(11,12)	False	(13)	