Assignment 1

Additional information: code requires **C++11**.

One of outputs of program:

Number of wins in A*: 99
Number of wins in BackTracking: 99
Number of fails in A*: 1
Number of fails in BackTracking: 1
Average steps in A*: 9
Average steps in Backtracking: 39
Average time until solution of A*: 20212 ns
Average time until solution of BackTracking: 5060 ns

Characteristics of environment for Red Riding Hood agent:

- **Partially observable (**because red riding hood does not know where placed wolf and bear, but she can explore environment and then know**)**
- **Deterministic (**outcome (win or fail) depends only on which steps red riding hood will do**)**
- **Discrete (**because in this game agent's percepts, actions and possible states of the environment are discrete**)**
- **Multiple Agent (**because we have a multiple agents and each does its work. Wolf and bear are competitive agents, because they contribute to the loss (each intersection with bear or its range makes red riding hood lose 2 points, wolf kills red riding hood as she comes in his detection range). Granny and wood cutter are cooperation agents, because they contribute to the win (if we in same cell with granny, we win and wood cutter give back red riding hood all points, if she lose some)**)**
- **Sequential (**because each step that red riding hood does affect on next steps. For example, in A* algorithm, we go to the cell, which most closely approximates the goal (granny), from that cell we can go to one that will approximate the goal better. Also we can add that red riding does not go to cells, in which she was and it means that visiting every cell approximate us to the goal(granny))

- **Static (**because environment does not change during game and environment stay the same until the end**)**
- **Known (**we know all rules of the environment**)**
- **Benign (**because environment does not want to make my life worse(like in chess game), it is just random environment**)**

Characteristics of Red Riding Hood agent:

1. Agent based on backtracking algorithm is Model-based reflex agent, because it remember cells, where it was and do not go there again, i.e. it remembers map of board. And it walks while it can (if neighbouring cell is not wolf range or board edge then go there) in the board () until it finds granny. And if it in situation, in which it can not go to next cell, it returns to the previous cell and try to go to neighbouring unvisited cells.

2. Agent based on A* algorithm is utility-based agent, because in every step it chooses cell to go, that will approximate to the goal(granny). Utility - is how I am close to granny. Before go to some cell, I calculate, how happy I will in this cell (how close I will be to granny). Only after this I choose cell, in which I will be most happy, and go there.

Characteristics of environment for Granny agent:

- **Partially observable (**because granny does not know where placed other agents**)**
- **Deterministic (**there is no randomness, Red Riding Hood can visit this agent and this agent will decide does she win or not**)**
- **Discrete (**because in this game agent's percepts, actions and possible states of the environment are discrete**)**
- **Multiple Agent (**because we have a multiple agents and each does its work**)**
- **Episodic (**because next actions of granny does not depend on current actions**)**
- **Static (**because environment does not change during game and environment stay the same until the end**)**

- **Known (**we know all rules of the environment**)**
- **Benign (**because environment does not want to make my life worse(like in chess game), it is just random environment**)**

Characteristics of Granny agent:
Granny agent is simple reflex agent, because she just waits to red riding hood visit her. If RRH visits her with 6 points, then she will say to her that she wins and get all points.

Characteristics of environment for Wood Cutter agent:

- **Partially observable (**because wood cutter does not know where placed other agents**)**
- **Deterministic (**there is no randomness, Red Riding Hood can visit this agent and this agent will give her points, if she has not all**)**
- **Discrete (**because in this game agent's percepts, actions and possible states of the environment are discrete**)**
- **Multiple Agent (**because we have a multiple agents and each does its work**)**
- **Episodic (**because next actions of wood cutter does not depend on current actions**)**
- **Static (**because environment does not change during game and environment stay the same until the end**)**
- **Known (**we know all rules of the environment**)**
- **Benign (**because environment does not want to make my life worse(like in chess game), it is just random environment**)**

Characteristics of Wood cutter agent:
Wood cutter agent is simple reflex agent, because she just waits to red riding hood visit him. If RRH visits him with less than 6 points, then he will give all her points back.

Characteristics of environment for Wolf agent:

- **Partially observable (**because wolf agent does not know where placed other agents**)**
- **Deterministic (**there is no randomness, Red Riding Hood can visit this agent or his range and this agent will kill her**)**
- **Discrete (**because in this game agent's percepts, actions and possible states of the environment are discrete**)**
- **Multiple Agent (**because we have a multiple agents and each does its work**)**
- **Episodic (**because next actions of wolf does not depend on current actions**)**
- **Static (**because environment does not change during game and environment stay the same until the end**)**
- **Known (**we know all rules of the environment**)**
- **Benign (**because environment does not want to make my life worse(like in chess game), it is just random environment**)**

Characteristics of Wolf agent:
Wolf agent is simple reflex agent, because it just waits to red riding hood visit him and his range. If RRH visits him, it will kill her.

Characteristics of environment for Bear agent:
- **Partially observable (**because bear agent does not know where placed other agents**)**
- **Deterministic (**there is no randomness, Red Riding Hood can visit this agent or his range and this agent will makes her lose 2 points**)**
- **Discrete (**because in this game agent's percepts, actions and possible states of the environment are discrete**)**
- **Multiple Agent (**because we have a multiple agents and each does its work**)**
- **Episodic (**because next actions of bear does not depend on current actions**)**
- **Static (**because environment does not change during game and environment stay the same until the end**)**
- **Known (**we know all rules of the environment**)**

- **Benign (**because environment does not want to make my life worse(like in chess game), it is just random environment**)**

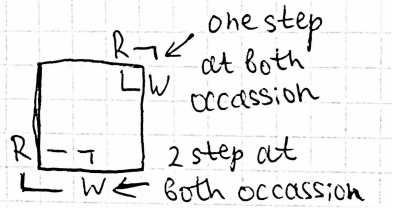
Characteristics of Bear agent:
Bear agent is simple reflex agent, because it just waits to red riding hood visit him or his range. If RRH visits him, it will makes her lose 2 points.

How works my backtracking algorithm:
I choose depth first search as backtracking algorithm. It just walks in the board while it can do it (while next neighbour cells are not wolf range or bear range or end board or visited cells). If it can not go to neighbour cells, then it returns one step back and try to go some cell, which it can go. Also I should say that every time that we are in some cell we mark it as visited, that we will not go there again.

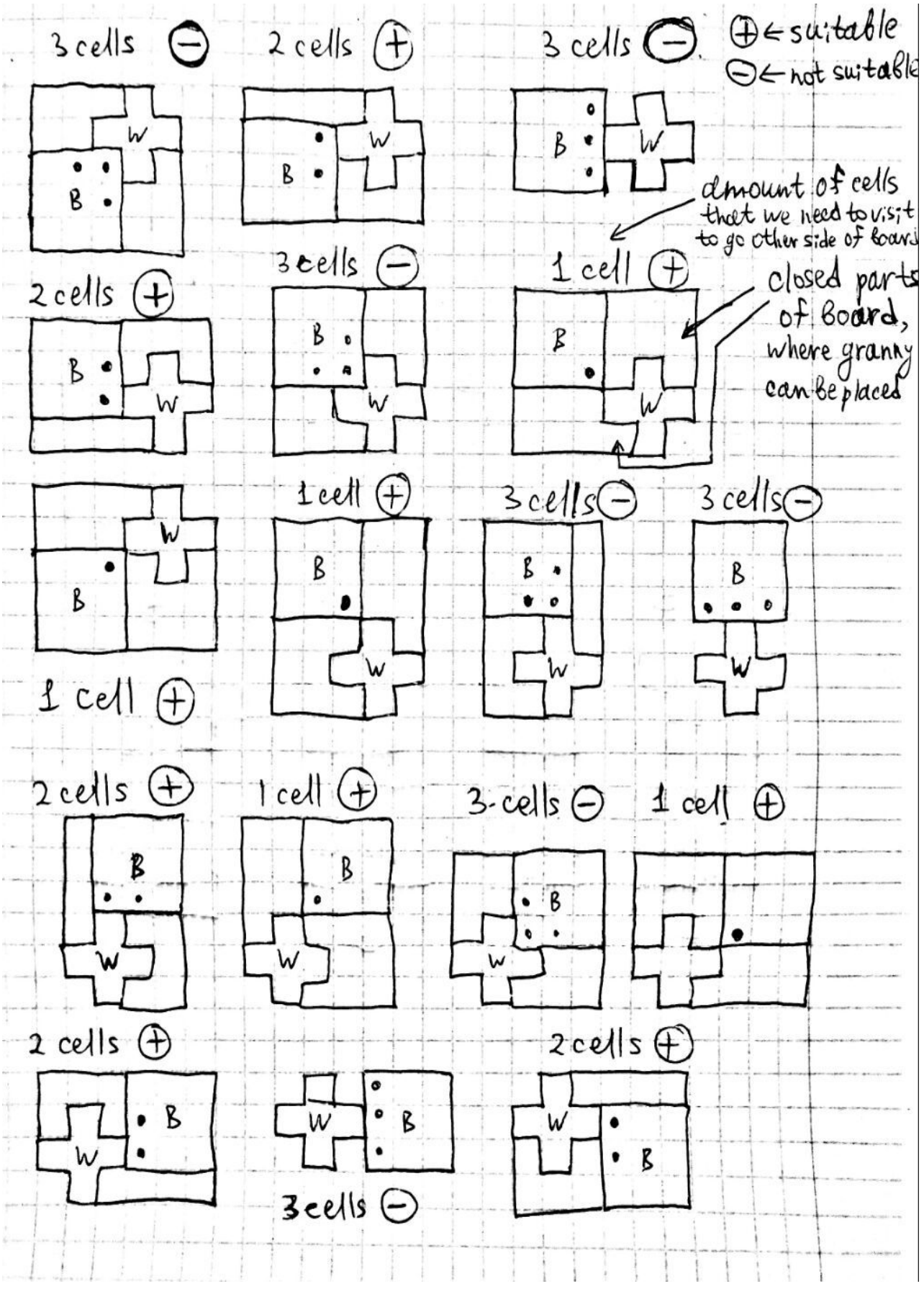
How works my A* algorithm:
I implemented default A* algorithm with cost function f = g + h, where g - number of cells we passed by from the start to get to some cell, h - square Euclidean distance between cell, where we want to go, and goal(granny). I took not just Euclidean distance, but square Euclidean distance, because easier to store them (we can use just integer, instead of doubles or floats) and our comparisons between cells will still be the same. I used priority_queue to store cells and choose best of them to go (with least cost). When I visit some cell, I look to unvisited neighbour cells, calculate cost function for them and put them into priority_queue. Also I should say that every time that we are in some cell we mark it as visited, that we will not go there again.

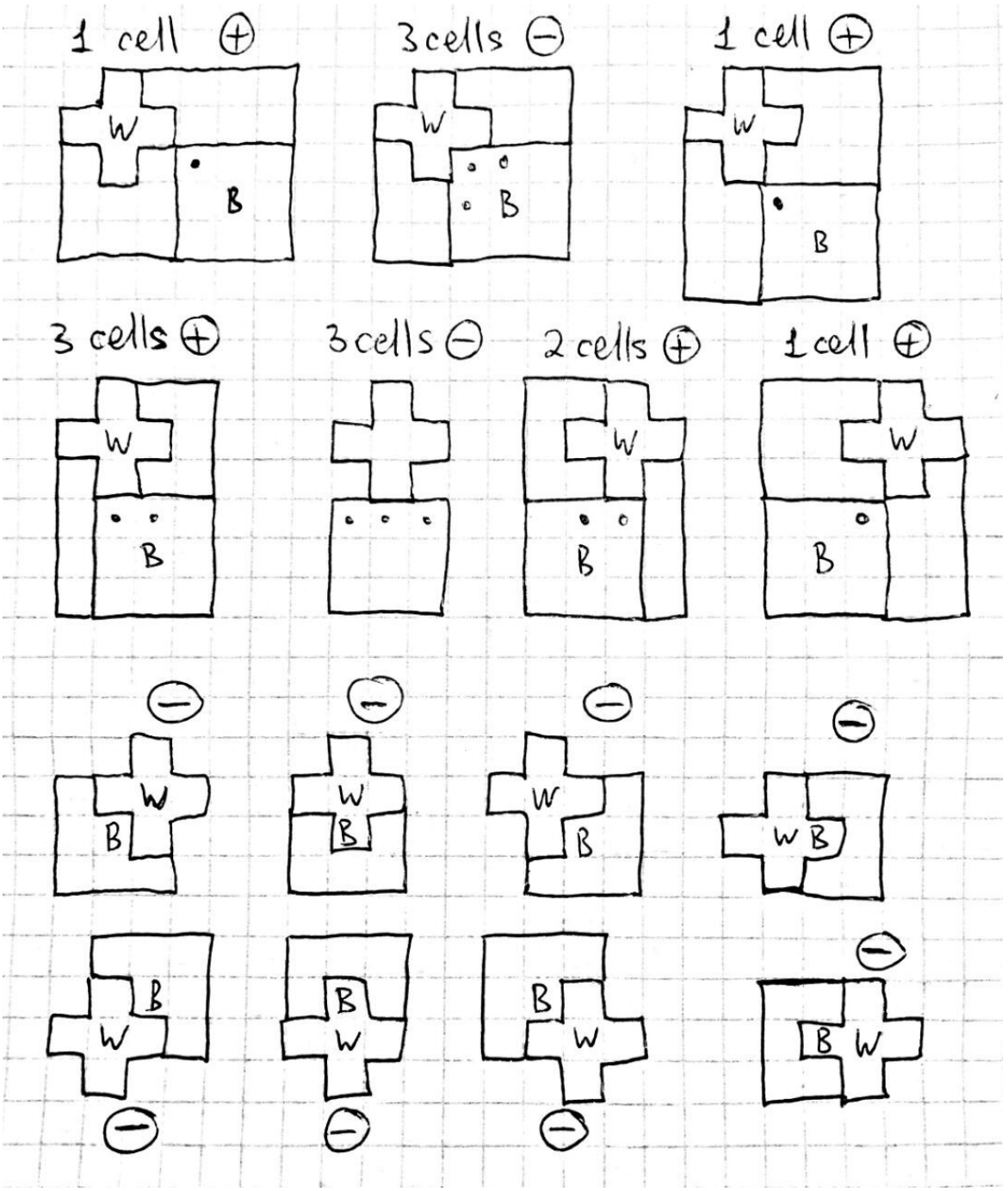
Optimizations, because of some difficulties:
First of all, in my implementation red riding hood agent never go to cell with range of wolf or with wolf, because then we will die. And initially, we can not go to cells with bear range or with bear. I should say that we can visit only 2 bear cells (bear and bear range) without fail then we should

go to wood cutter and get all points back. Red riding hood agent can not shorten the distance to granny, because she can pass only 2 bear cells (bear or bear agent), the best example with wood cutter near, but also can be without wood cutter near:

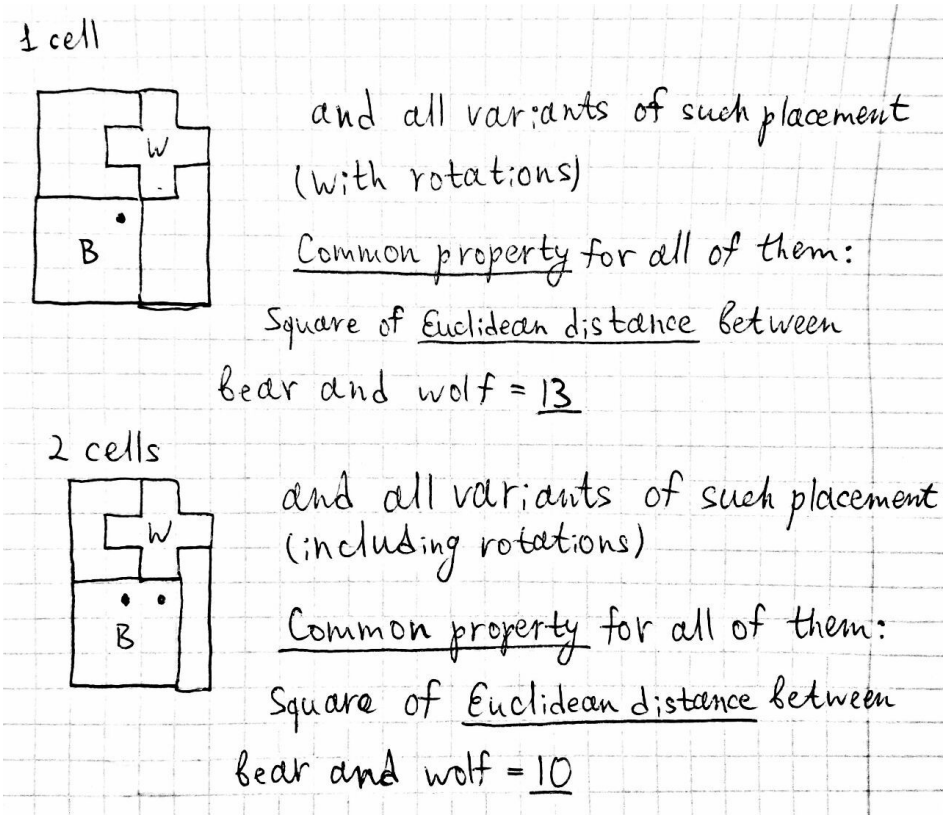


It means that we should go to bear cell (bear and bear range) only when we did not find granny at available set of cells. But if we did not find granny then it can mean a lot of occasions. We can visit only 1 or 2 bear cells and can not visit wolf cells. It means that we can go to granny, which is closed by wolf range and bear range from us. All variants of how wolf and bear can be placed, that their range will close some side of board (continue on next page):


3 cells ⊖
2 cells ⊕
3 cells ⊖
⊕ ← suitable
⊖ ← not suitable
W
B
amount of cells that we need to visit to go other side of board
1 cell ⊕
closed parts of board, where granny can be placed
2 cells ⊕
3 cells ⊖
1 cell ⊕
3 cells ⊖
3 cells ⊖
1 cell ⊕
2 cells ⊕
1 cell ⊕
3 cells ⊖
1 cell ⊕
2 cells ⊕
2 cells ⊕
3 cells ⊖


1 cell ⊕
3 cells ⊖
1 cell ⊕
3 cells ⊕
3 cells ⊖
2 cells ⊕
1 cell ⊕
W
B

We can pass to closed by wolf range and bear range side of board without fail only in this cases:



And if Red Riding Hood agent will visit and can not find granny then I check placement of bear and wolf and if it is like that I have shown before then I mark cells of bear, to which I can go, as unvisited cells and go there. If wood cutter will be there then Red Riding Hood can get all points back and win, otherwise she will fail. In my implementation of A* algorithm every cell has a default cost. Initially all of them equals to 0. Wolf and his range cells will mark as visited. Then I preprocess board and if I found placement of bear and wolf such I have shown before, which allows me to go to closed side of board then I mark this cells of bear range like unvisited and assign their cost to 100, it means that before going to this cell, we will visit all available cells. In implementation of backtracking algorithm, if I executed dfs and red riding hood did not win then I check if we can pass to closed part of board, and if can then mark as unvisited this cells of range bear. But there can be one problem: we can allow red riding hood to another side of board, but she can find wood cutter, get points back, but then did not visit granny, because she

or path to her can be already visited. And in this occasions, I mark as unvisited all cells that we visit after going to closed part and run algorithm again.

Agent Red Riding Hood in my implementation will fail only in cases when she can not go physically to granny with 6 points. Reasons can be following: she closed by bear or wolf and we can not go to her without failing, because we need to visit 3 or more bear cells (bear and his range) or we need to visit wolf cell. It can be look like this: