1. Descrição do Problema

O apocalipse zumbi começou. Grandes hordas de mortos-vivos, chamados "caminhantes (walkers)" ou "mordedores (biters)", devoram pessoas e sua mordida é infecciosa para os seres humanos. Os sobreviventes têm conhecimentos limitados sobre o que realmente está acontecendo no mundo. Um grupo enfrenta dilemas na luta para manter-se vivo. Os sentimentos confusos e os desafios do dia-a-dia em um mundo hostil e praticamente dominado por mortos-vivos são constantes e perturbadores.

Rick, um xerife de uma pequena cidade, luta para manter seu grupo vivo e longe da ameaça zumbi. Juntos eles precisam adquirir novos meios apropriados de convívio social agora que as estruturas da sociedade entraram em colapso e a realidade se tornou atípica.

Durante a constante batalha, os sobreviventes peregrinam em busca de abrigos seguros, até que encontram proteção atrás das grades de uma prisão abandonada. Porém a paz conquistada não dura muito tempo. Um grande bando de zumbis está prestes a chegar às mediações da prisão e a estrutura de muros não vai aguentar por muito tempo.

O seu objetivo é ajudar Rick a reunir seu grupo e escapar da prisão o mais rápido possível.

Figura 1. Carl, Rick, Daryl, Glen e Maggie.

2. Implementação

O trabalho prático consiste em implementar um agente capaz de locomover-se autonomamente pela **Prisão**, explorar os diversos ambientes e reunir o grupo de sobreviventes para a fuga. Para isso, você deve utilizar o **algoritmo de busca heurística A***.

O agente deve ser capaz de calcular automaticamente a **melhor rota** para que **Rick possa encontrar Carl, Daryl, Glen e Maggie** e, depois, **guiá-los para sair pelo portão da prisão**.

O mapa da **Prisão** é mostrado na Figura 2.

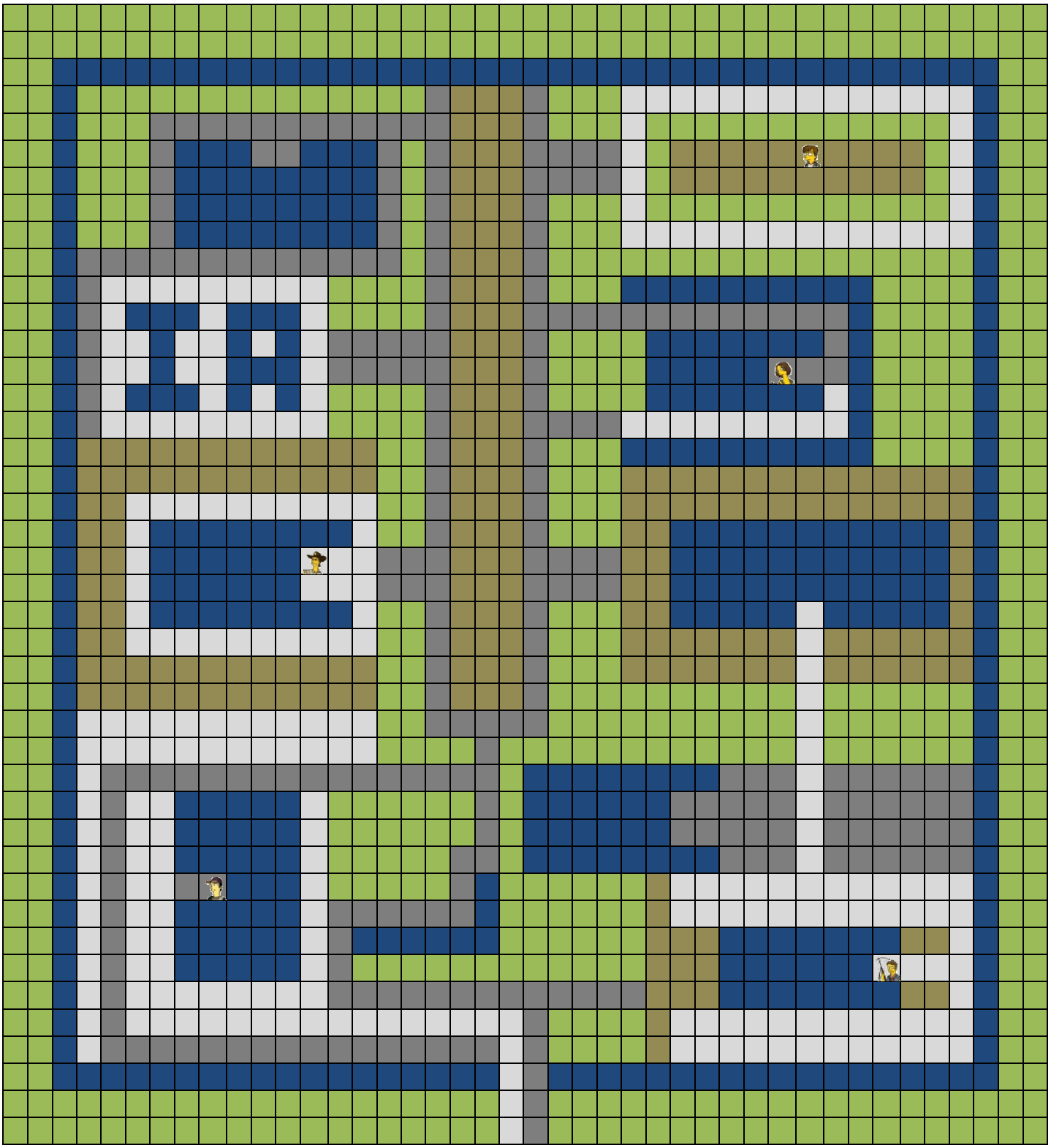

Figura 2. Mapa da Prisão.

A **Prisão** é formada por **5 tipos de terrenos**: asfalto (região cinza escuro), grama (região verde), terra (região marrom), paralelepípedo (região cinza claro) e edifícios (região azul).

Para piorar a situação, Rick foi atingido na perna durante a última batalha e tem dificuldades para caminhar. Dessa forma, cada tipo de terreno exige uma determinada quantidade de esforço, o que faz com que Rick caminhe com mais ou menos dificuldade, dependendo do tipo de terreno.

Para passar por cada tipo de terreno, Rick gasta uma determinada quantidade de tempo:

- **Asfalto** – Custo: +1
- **Terra** – Custo: +3
- **Grama** – Custo: +5
- **Paralelepípedo** – Custo: +10

Rick **nunca pode passar por regiões de edifícios** (regiões de cor azul no mapa da Figura 2).

As localizações dos 4 membros do grupo estão definidas na Figura 2. Rick inicia sua jornada no **Refeitório (posição [21,13] no mapa)** e termina após **reunir todo o grupo** e **sair da prisão pela porta (parte inferior do mapa)**. A melhor rota para cumprir essa missão é a rota de menor custo levando em consideração o terreno.

3. Informações Adicionais

- O mapa principal deve ser representado por uma matriz 42 x 42 (igual à mostrada na Figura 2).

- O agente sempre **inicia** a jornada no **Refeitório** (ponto onde Rick está no mapa [21, 13]).

- O agente sempre **termina** a sua jornada ao **reunir todo o grupo** e **chegar na porta de saída da prisão (parte inferior do mapa)**.

- O agente não pode andar na diagonal, somente na **vertical** e na **horizontal**.

- Rick pode reunir o grupo **em qualquer ordem**. Porém, ordens diferentes vão resultar em custos totais diferentes.

- Deve existir uma maneira de **visualizar os movimentos** do agente, mesmo que a interface seja bem simples. Podendo até mesmo ser uma matriz desenhada e atualizada no console.

- **Os mapas devem ser configuráveis**, ou seja, deve ser possível modificar o tipo de terreno em cada local. O mapa pode ser lido de um arquivo de texto ou deve ser facilmente editável no código.

- O programa deve **exibir o custo do caminho** percorrido pelo agente enquanto ele se movimenta pelo mapa e também o **custo final** ao terminar a execução.

- O programa pode ser implementado em **qualquer linguagem**.

4. Dicas

- Note que este problema é semelhante ao problema do **Caixeiro Viajante**. É necessário encontrar a melhor rota para visitar todos os membros do grupo uma única vez. No trabalho não é obrigatória a resolução deste problema, mas é única maneira de garantir o melhor custo.

- Implemente a função de busca de uma forma genérica, pois será necessário executa-la múltiplas vezes para diferentes destinos.

5. Extra

a) A interface gráfica não é o objetivo desse trabalho, mas quem implementar uma “boa” interface gráfica (2D ou 3D) para representar o ambiente e o agente receberá até 2 pontos extras na nota.

b) O programa que conseguir resolver o problema proposto **com o menor custo**, receberá **1 ponto extra** na nota. Em caso de empate, o critério de desempate será a velocidade de execução do algoritmo de busca. Nesse caso, o trabalho deverá ter implementado um mecanismo para calcular o tempo gasto pelo algoritmo.

6. Orientações

- Data de entrega: **06/03/2016**

- O trabalho pode ser feito **individualmente** ou em **grupos** de no **máximo 2 pessoas**.

- Na data especificada, cada grupo deverá entregar um arquivo comprimido (ZIP ou RAR) contendo todo o código. A nomenclatura deverá seguir o seguinte padrão: <NomeEUltimoSobrenomeDoAluno1>_<NomeEUltimoSobrenomeDoAluno2>_<Trabalho_Busca_Heuristica>. Ao digitar seu nome os sinais '<' e '>' devem ser eliminados. Apenas o primeiro e o último nome devem compor o nome do arquivo.
    - Ex: JoseSilva_MariaOliveira_Trabalho_Busca_Heuristica.rar

- Em caso de atraso na entrega, o valor em pontos de cada atividade será recalculado de acordo com a seguinte fórmula:

$$valor_{com\ atraso} = valor_{inicial} * (1 - Dias_{atraso} * 0{,}08)$$