UB Social

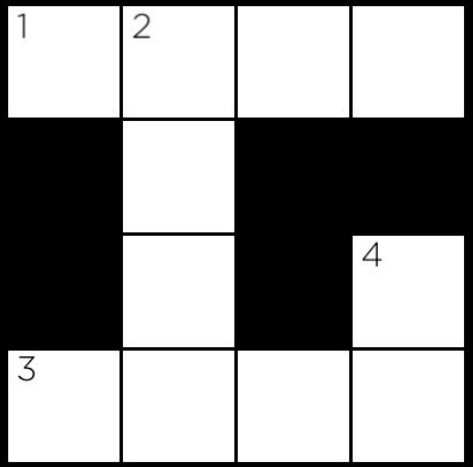
CS50AI with Python

7.Crossword

Mateus Schwede

Problemática

- Escreva uma IA para gerar palavras cruzadas;
- Dada estrutura de jogo de palavras cruzadas (ou seja, quais quadrados da grade devem ser preenchidos com uma letra) e lista de palavras a serem usadas, deve-se escolher quais palavras devem entrar em cada sequência vertical ou horizontal de quadrados.
- Pode-se modelar esse tipo de problema como problema de satisfação de restrições. Cada sequência de quadrados é uma variável, para a qual precisa-se decidir seu valor (qual palavra no domínio de palavras possíveis preencherá essa sequência)



Problemática

- Na estrutura, há 4 variáveis, representando as 4 palavras para preencher na palavra cruzada (cada uma indicada por nº na imagem);
- Cada variável é definida por 4 valores: a linha em que começa (seu valor i), a coluna em que começa (seu valor j), a direção da palavra (para baixo ou horizontalmente) e o comprimento da palavra;
- A variável 1, por exemplo, é variável representada por linha de 1 (assumindo 0 indexado a partir do topo), coluna de 1 (também assumindo 0 indexado a partir da esquerda), direção horizontalmente e comprimento de 4;
- Como em muitos problemas de satisfação de restrições, essas variáveis têm restrições unárias e binárias;
- Restrição unária em variável é dada por seu comprimento. Para a Variável 1, por exemplo, valor BYTE satisfaria a restrição unária, mas valor BIT não (possui nº errado de letras);
- Quaisquer valores que não satisfaçam restrições unárias de variável podem, portanto, ser removidos do domínio da variável imediatamente.

Problemática

- Restrições binárias em variável são dadas por sua sobreposição com variáveis vizinhas;
- Variável 1 tem única vizinha: variável 2. Variável 2 tem 2 vizinhas: variável 1 e 3;
- Para cada par de variáveis vizinhas, essas variáveis compartilham sobreposição: quadrado único comum a ambas;
- Representa-se essa sobreposição como índice do caractere na palavra de cada variável que deve ser mesmo caractere;
- Exemplo, sobreposição entre variável 1 e 2 pode ser representada como par (1, 0), significando que caractere da variável 1 no índice 1 deve ser mesmo que caractere da variável 2 no índice 0 (assumindo indexação 0);
- Sobreposição entre variável 2 e 3 seria, portanto, representada como par (3, 1): caractere 3 do valor da variável 2 deve ser mesmo que caractere 1 do valor da variável 3;
- Para este problema, adicionaremos a restrição adicional de que todas as palavras devem ser diferentes: a mesma palavra não deve ser repetida várias vezes no quebra-cabeça;
- Desafio é escrever programa para encontrar atribuição satisfatória: palavra diferente (de lista de vocabulário fornecida) para cada variável, de modo que todas restrições unárias e binárias sejam atendidas.

Instruções

- Baixe código de https://cdn.cs50.net/ai/2023/x/projects/3/crossword.zip e descompacte-o.

Funcionamento

- Há 2 arquivos Python neste projeto: crossword.py e generate.py. O 1º foi escrito inteiramente para você, enquanto o 2º contém algumas funções que para implementar;
- crossword.py: Este arquivo define 2 classes: Variável (represeta variável em jogo de palavras cruzadas) e Palavras Cruzadas (representa o jogo);
- Para criar variável, especifica-se 4 valores: linha i, coluna j, direção (constante Variable.ACROSS ou constante Variable.DOWN) e comprimento;
- Classe Crossword requer 2 valores para criar jogo de palavras cruzadas: structure_file que define estrutura do jogo (o _ é usado para representar células em branco, qualquer outro caractere representa células que não serão preenchidas) e words_file que define lista de palavras (1 em cada linha) a serem usadas no vocabulário do jogo.

Funcionamento

- Para qualquer objeto crossword, armazena-se os valores:
  - crossword.height: inteiro que representa altura das palavras cruzadas;
  - crossword.width: inteiro que representa largura das palavras cruzadas;
  - crossword.structure: lista 2D que representa estrutura das palavras cruzadas. Para qualquer linha i e coluna j válidas, crossword.structure[i][j] será True se célula estiver em branco (1 caractere deve ser preenchido) e será False caso contrário (nenhum caractere deve ser preenchido nessa célula);
  - crossword.words: conjunto de todas palavras utilizadas na construção das palavras cruzadas;
  - crossword.variables: conjunto de todas variáveis nas palavras cruzadas (cada uma é objeto Variable);
  - crossword.overlaps: dicionário que mapeia par de variáveis para sua sobreposição. Para quaisquer 2 variáveis distintas v1 e v2, crossword.overlaps[v1, v2] será None se 2 variáveis não se sobrepuserem e será par de inteiros (i, j) se variáveis se sobrepuserem. Par (i, j) deve ser interpretado como significando que o i-ésimo caractere do valor de v1 deve ser memso que j-ésimo caractere do valor de v2.

Funcionamento

- Objetos de palavras cruzadas também suportam método neighbors que retorna todas variáveis que se sobrepõem a determinada variável. Ou seja, crossword.neighbors(v1) retornará conjunto de todas variáveis vizinhas da variável v1;
- generate.py: classe CrosswordCreator será utilizada resolver as palavras cruzadas. Quando objeto CrosswordCreator é criado, ele recebe propriedade crossword que deveria ser objeto Crossword (e, portanto, possui todas propriedades descritas acima);
- Cada objeto CrosswordCreator também recebe propriedade domains: dicionário que mapeia variáveis para conjunto de palavras possíveis que variável pode assumir como valor;
- Inicialmente, esse conjunto de palavras são todas palavras do vocabulário, mas criam-se funções para restringir tais domínios.

Funcionamento

- Função print imprimirá no terminal representação do quebra-cabeça de palavras cruzadas para determinada tarefa (cada tarefa, nesta função e em outros lugares, é dicionário que mapeia variáveis para suas palavras correspondentes)
- Função save gerará arquivo de imagem correspondente a determinada tarefa (necessário instalar Pillow com pip3);
- Função letter_grid é função auxiliar usada por print e save que gera lista 2D de todos caracteres em suas posições apropriadas para determinada tarefa;
- Função solve faz três coisas: 1º, chama enforce_node_consistency para impor consistência dos nós nas palavras cruzadas, garantindo que todos valores no domínio de variável satisfaçam restrições unárias. Em seguida, chama ac3 para impor consistência dos arcos, garantindo que restrições binárias sejam satisfeitas. Por fim, chama backtrack em atribuição inicialmente vazia (dicionário vazio dict()) para tentar calcular solução ao problema;
- Funções enforce_node_consistency, ac3 e backtrack, no entanto, precisam set implementadas (entre outras funções).

Especificações

- Conclua implementação de enforce_node_consistency, revise, ac3, assignment_complete, consistent, order_domain_values, selected_unassigned_variable e backtrack em generate.py para que IA gere palavras cruzadas completas, se possível;
- Função enforce_node_consistency deve atualizar self.domains de forma que cada variável seja consistente com os nós:
    - Consistência dos nós é alcançada quando, para cada variável, cada valor em seu domínio é consistente com restrições unárias da variável. Em palavras cruzadas, significa garantir que cada valor no domínio de variável tenha mesmo nº de letras que comprimento da variável;
    - Para remover valor x do domínio de variável v, como self.domains é dicionário que mapeia variáveis para conjuntos de valores, pode-se chamar self.domains[v].remove(x);
    - Nenhum valor de retorno é necessário para esta função.

Especificações

- Função revise deve tornar variável x consistente com variável y:
  - x e y serão ambos objetos Variable representando variáveis no quebra-cabeça;
  - x é consistente com y quando cada valor no domínio de x tem valor possível no domínio de y que não causa conflito;
  - Conflito, no contexto das palavras cruzadas, é quadrado para qual 2 variáveis discordam sobre qual valor de caractere ele deve assumir;
  - Para tornar x consistente com y, remove-se qualquer valor do domínio de x que não tenha valor possível correspondente no domínio de y;
  - Pode-se acessar self.crossword.overlaps para obter sobreposição, se houver, entre 2 variáveis;
  - Domínio de y deve ser deixado inalterado;
  - Função deve retornar True se revisão foi feita no domínio de x; retornar False se nenhuma revisão foi feita.

Especificações

- Função ac3 deve, usando algoritmo AC3, impor consistência de arcos no problema:
  - Consistência de arcos é alcançada quando todos valores no domínio de cada variável satisfazem as restrições binárias dessa variável;
  - Algoritmo AC3 mantém fila de arcos para processar;
  - Função recebe argumento opcional arcs, representando lista inicial de arcos a serem processados;
  - Se arcs for None, função deve começar com fila inicial de todos arcos do problema. Caso contrário, incia-se com fila inicial apenas dos arcos que estão na lista arcs (onde cada arco é tupla (x, y) de variável x e variável diferente y);
  - Para implementar AC3, revisa-se cada arco na fila, um de cada vez. Porém, sempre que fizer alteração em domínio, poderá ser necessário adicionar arcos adicionais à fila para garantir que outros arcos permaneçam consistentes;

Especificações

- Pode-se chamar função revise na implementação de ac3;
- Se, no processo de aplicação da consistência do arco, remover todos valores restantes de domínio, retorne False (significa que é impossível resolver o problema, pois não há mais valores possíveis para variável). Caso contrário, retorne True;
- Dispensa-se aplicar unicidade das palavras nesta função (verificação será na função consistent).

Especificações

- Função assignment_complete deve verificar se determinada atribuição foi concluída:
  - assignment é dicionário cujas chaves são objetos Variable e valores são strings que representam palavras que essas variáveis assumirão;
  - assignment é considerada concluída se cada variável de palavras cruzadas for atribuída a valor;
  - Função deve retornar True se atribuição for concluída e False caso contrário.

Especificações

- Função consistent deve verificar se determinada atribuição é consistente:
  - assignment é dicionário em que chaves são objetos Variable e valores são strings que representam palavras que essas variáveis assumirão;
  - assignment pode não ser completa: nem todas variáveis estarão necessariamente presentes na atribuição;
  - assignment é consistente se satisfazer todas restrições do problema: ou seja, todos valores são distintos, cada valor tem comprimento correto e não há conflitos entre variáveis vizinhas;
  - Função deve retornar True se atribuição for consistente e False caso contrário.

Especificações

- Função order_domain_values deve retornar lista de todos valores no domínio de var, ordenados conforme heurística dos valores menos restritivos:
    - var será objeto Variable, representando variável no quebra-cabeça;
    - Heurística dos valores menos restritivos é calculada como nº de valores descartados para variáveis vizinhas não atribuídas. Ou seja, se atribuir var a valor específico resultar na eliminação de n escolhas possíveis para variáveis vizinhas, ordena-se resultados em ordem crescente de n;
    - Qualquer variável presente na atribuição já possui valor e, portanto, não deve ser contada ao calcular nº de valores descartados para variáveis vizinhas não atribuídas;
    - Para valores de domínio que eliminam mesmo nº de escolhas possíveis para variáveis vizinhas, qualquer ordenação é aceitável;
    - Pode-se acessar self.crossword.overlaps para obter sobreposição, se houver, entre 2 variáveis;
    - Pode-se implementar esta função primeiro retornando lista de valores em qualquer ordem arbitrária (o que ainda deve gerar palavras cruzadas corretas). Com algoritmo funcionando, retorna e garante que valores sejam retornados na ordem correta;
    - Considera-se ordenar lista conforme chave específica.

Especificações

- Função select_unassigned_variable deve retornar única variável nas palavras cruzadas que ainda não foi atribuída por atribuição, conforme heurística do valor mínimo restante e, em seguida, heurística do grau:
    - assignment é dicionário onde chaves são objetos Variable e valores são strings que representam palavras que essas variáveis assumirão. Presume-se que atribuição não será completa: nem todas variáveis estarão presentes na atribuição;
    - Função deve retornar objeto Variable. Deve-se retornar variável com menor nº de valores restantes no domínio;
    - Se houver empate entre variáveis, escolhe-se entre aquelas com maior grau (mior nº de vizinhos);
    - Se houver empate em ambos casos, escolhe-se arbitrariamente entre variáveis empatadas;
    - Considera-se implementar 1º esta função retornando qualquer variável arbitrária não atribuída (gerando palavras cruzadas corretas);
    - Com algoritmo funcionando, retoma-se e garante que está retornando variável conforme heurística;
    - Considera-se classificar lista conforme chave específica.

Especificações

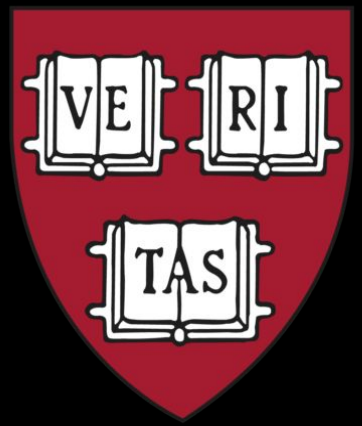
- Função backtrack deve aceitar atribuição parcial como entrada e, via busca por retrocesso, retornar atribuição completa e satisfatória de variáveis a valores, se possível:
    - assignment é dicionário onde chaves são objetos Variable e valores são strings que representam palavras que essas variáveis assumirão;
    - Atribuição de entrada pode não ser completa (nem todas variáveis necessariamente terão valores);
    - Caso pode-se gerar jogo de palavras cruzadas satisfatório, deve retornar atribuição completa: dicionário onde cada variável é chave e valor é palavra que variável deve assumir;
    - Se nenhuma atribuição satisfatória for possível, função deve retornar None;
    - Algoritmo, opcionalmente, seja mais eficiente se intercalar busca com inferência (mantendo consistência do arco sempre que fizer nova atribuição);
    - Por isso que função ac3 permite argumento arcs, caso deseja iniciar fila diferente de arcos.

Especificações

- Não pode-se modificar generate.py além das funções que especificação solicita, embora possa escrever funções adicionais e/ou importar outros módulos da biblioteca padrão do Python;
- Você pode-se modificar crossword.py.

Submissão

- Visual Studio Code online: https://cs50.dev
- Testar precisão da lógica do algoritmo: check50 ai50/projects/2024/x/crossword
- Testar estilização do código: style50 generate.py
- Para submissão:
  - Em https://submit.cs50.io/invites/d03c31aef1984c29b5e7b268c3a87b7b, entre com GitHub e autorize CS50;
  - Instale pacote Git, Python 3 (e pip), instalando pacotes: pip3 install style50 check50 submit50
  - Submeta o projeto: submit50 ai50/projects/2024/x/crossword
- Verificar avaliação: https://cs50.me/cs50ai.



UB Social

Mateus Schwede
HBS ID 202400167108 - DCE ID @00963203