از آنجایی که در هر لحظه $O(n)$ خانه در این بازی پر شده اند، تنها خانه هایی که مقدار آن ها غیر صفر است را ذخیره می کنیم و با این کار می توانیم دنیای بازی را در $O(n)$ واحد حافظه ذخیره کنیم. با توجه به اینکه برای عملیات های این بازی به ساختار و الگوریتم های کارا نیازمندیم ، می توانیم از درخت های خود متوازن استفاده کنیم که در این سوال، استفاده از درخت AVL با توجه به شرایط مسئله مناسب است. در این ساختار با توجه به متوازن بودن آن، کارایی الگوریتم های جستجو، حذف و اضافه کردن، $O(logn)$ خواهد بود.

برای ذخیره سازی این دنیای سه بعدی که از مولفه های i, j, k تشکیل شده است، کافی است به هر یک از این خانه ها یک عدد یکتا نسبت دهیم تا بتوانیم با استفاده از آن، درخت AVL مورد بحث را تشکیل دهیم. اگر فرایند اندیس گذاری را از خانه $(1, 1, 1)$ شروع کنیم و در راستای محور x حرکت کنیم، خواهیم داشت :

$$index(1,1,1) = 1, index(2,1,1) = 2, ..., index(n,1,1) = n$$

سپس یک واحد به مولفه j اضافه کرده و این روند را تکرار می کنیم. برای خانه هایی که مولفه j آنها 2 است داریم:

$$index(1,2,1) = 1 + n, index(2,2,2) = 2 + n, \ldots, index(n,2,2) = n + n$$

با تکرار این روند و پوشش کامل خانه هایی که مولفه k آنها 1 و تعداد آنها n^2 است، k را یک واحد افزایش داده و این روند را تا انتها تکرار می کنیم. بنابراین اندیس هر خانه را می توانیم به صورت تابعی از مولفه ها بنویسیم :

$$index(i,j,k) = i + n(j-1) + n^2(k-1)$$

بدین ترتیب با مجموعه اندیس و مقدار خانه های غیر صفر درخت دودویی مورد نظر را می سازیم که عملیات های آن به صورت زیر تعریف شده اند:

- تعیین مقدار هر خانه: این کار را با جستجو در درخت انجام می دهیم و درصورت ناموفق بودن جستجو مقدار آن خانه را صفر برمی گردانیم
- صفر کردن مقدار خانه: با حذف کردن گره متناظر در درخت، مقدار خانه را صفر می کنیم.
- اضافه کردن امتیاز: ابتدا اندیس خانه را جستجو کرده و در صورت موفق بودن، مقدار آن را 100 واحد افزایش می دهیم در غیر این صورت یک گره به درخت اضافه می کنیم.