علیرضا کمرزردی و امین امیری

داکیومنت پروژه ارضای محدودیت درس هوش مصنوعی

**فاز یک**

* **Hall**
* **CSP**

**فاز دو**

* **LCV**

ما در تابع ابتکاری lcv به دنبال مقدار هایی از دامنه هستیم که کمترین تداخل را با مقدار های دامنه نود های همسایه دارد. در واقع این تابع لیست مرتب شده از مقادیر دامنه بر اساس کمترین تداخل بصورت صعودی به ما میدهد.

def lcv(hall: Hall, assignment) -> list:  
 return sorted(hall.domain, key=lambda value: number\_of\_conflicts(hall, value, assignment))

* **MRV**
* **Forward\_checking**
* **backtracking**

**فاز سه**

* **AC3**

def ac3(csp: CSP) -> bool:  
 queue = list(csp.constraints)  
  
 while len(queue) != 0:  
 i, j = queue.pop(0)  
 if revise(csp, i, j):  
 if len(csp.halls[i].domain) == 0:  
 return False  
 for k in (csp.halls[i].constraint - {j}):  
 if (k, i) not in queue:  
 queue.append((k, i))  
  
 return True  
  
  
def revise(csp, i, j) -> bool:  
 revised = False  
  
 for value in csp.halls[i].domain.copy():  
 if len(csp.halls[j].domain - {value}) == 0:  
 csp.halls[i].domain.remove(value)  
 revised = True  
  
 return revised