Druga domaća zadaća

Igra Connect4

1. Implementacija

Upute: U ovom odjeljku potrebno je opisati ključne dijelove funkcionalnosti koristeći isječke programa i snimke zaslona. Obavezno uključite sljedeće elemente s odgovarajućim komentarima:

- *Isječak programa koji prikazuje pripremu poslova na glavnom (master) procesu.*
- *Isječke programa koji pokazuju kako se zadaci prenose s glavnog (master) procesa na radničke (worker) procese.*
- *Snimku zaslona koja prikazuje posljednja dva koraka igre u kojoj računalo pobjeđuje.*

Priprema poslova na glavnom procesu:

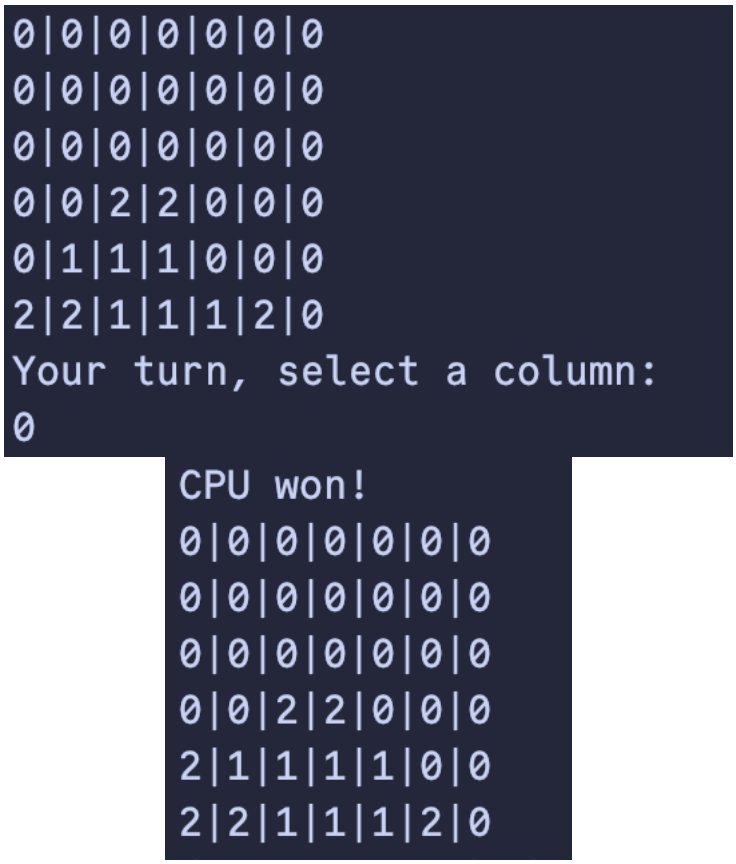
```
def create_tasks(board: Board, depth: int):
    current_player = CPU

    tasks = [Task( id: 0,  last_col_played: 0, current_player, board)]

    for level in range(depth):
        level_tasks = []

        for idx, task in enumerate(tasks):
            for col in range(BOARD_WIDTH):
                new_board = task.board.copy()
                if not new_board.move_legal(col):
                    continue

                task_id = idx * 7 + col

                new_board.make_move(col, current_player)
                level_tasks.append(Task(task_id, col, get_opponent(current_player), new_board))

        tasks = level_tasks

        current_player = HUMAN if current_player == CPU else CPU

    return tasks
```

Prenošenje taskova na workere:

```python
tasks = create_tasks(board, MASTER_DEPTH)
expected_completed_tasks = len(tasks)
completed_tasks = []

# send initial tasks
for worker in range(1, mpi_size):
    if tasks:
        task = tasks.pop(0)
        mpi_comm.send(obj=task, dest=worker, tag=WORK_TAG)

# receive from workers and send remaining tasks
while tasks:
    result = mpi_comm.recv(source=MPI.ANY_SOURCE, status=mpi_status, tag=COMPLETED_TAG)
    completed_tasks.append(result)
    new_task = tasks.pop(0)
    mpi_comm.send(obj=new_task, dest=mpi_status.source, tag=WORK_TAG)

# collect remaining tasks
while len(completed_tasks) != expected_completed_tasks:
    result = mpi_comm.recv(source=MPI.ANY_SOURCE, tag=COMPLETED_TAG)
    completed_tasks.append(result)

print("Received all tasks")

best_move = process_results(completed_tasks, MASTER_DEPTH)
```

```python
# workers
else:
    while True:
        task = mpi_comm.recv(source=0, tag=WORK_TAG)

        if task == "kill":
            break

        result = process_task(task, MAX_DEPTH)
        mpi_comm.send(obj=result, dest=0, tag=COMPLETED_TAG)
```

Posljednja dva poteza prije pobjede CPU-a:

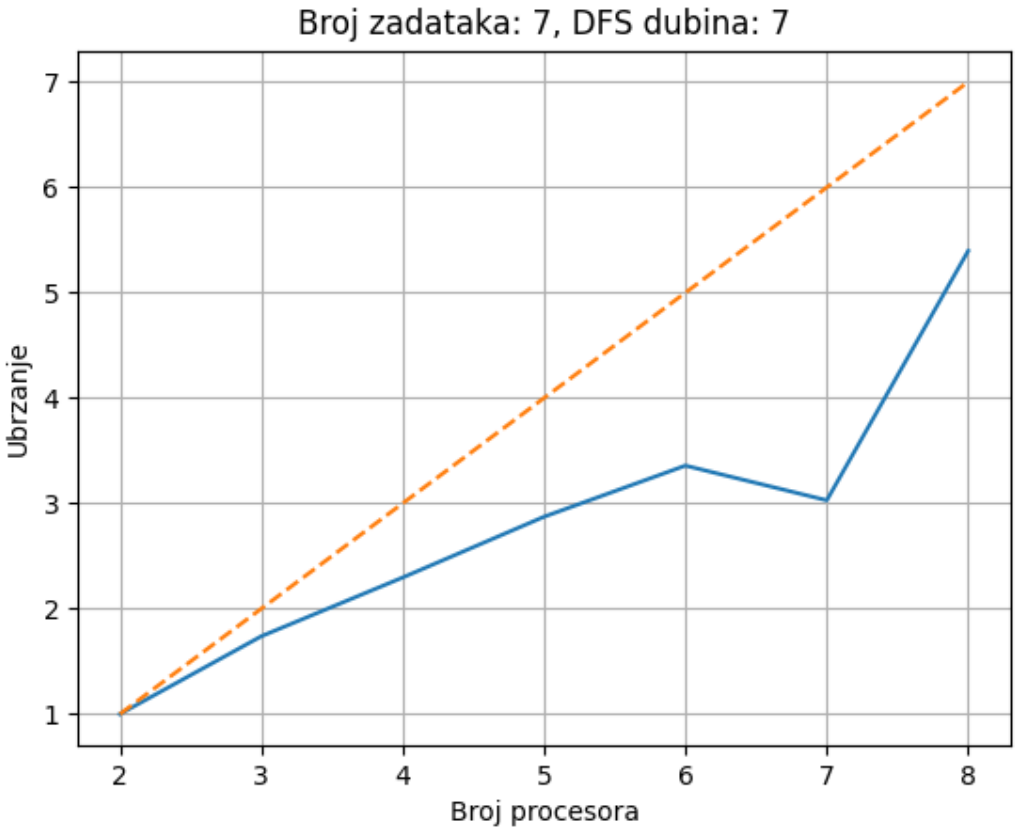


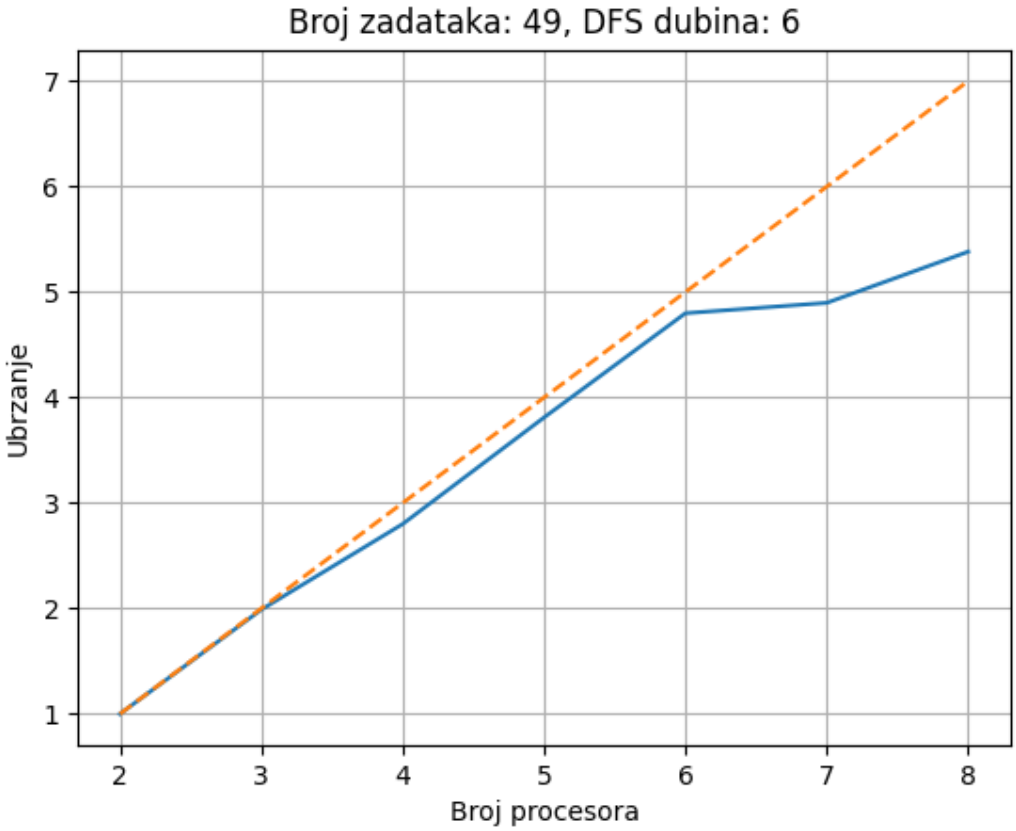
CPU je broj 1, player je broj 2.

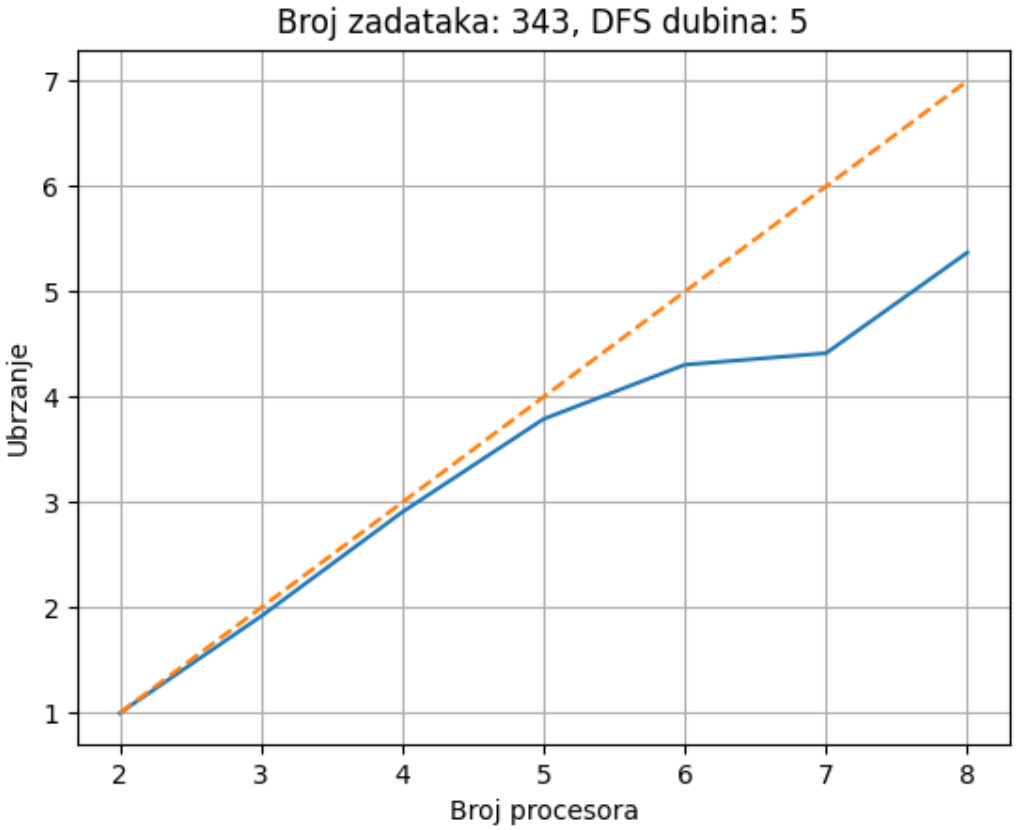
2. Kvantitativna analiza

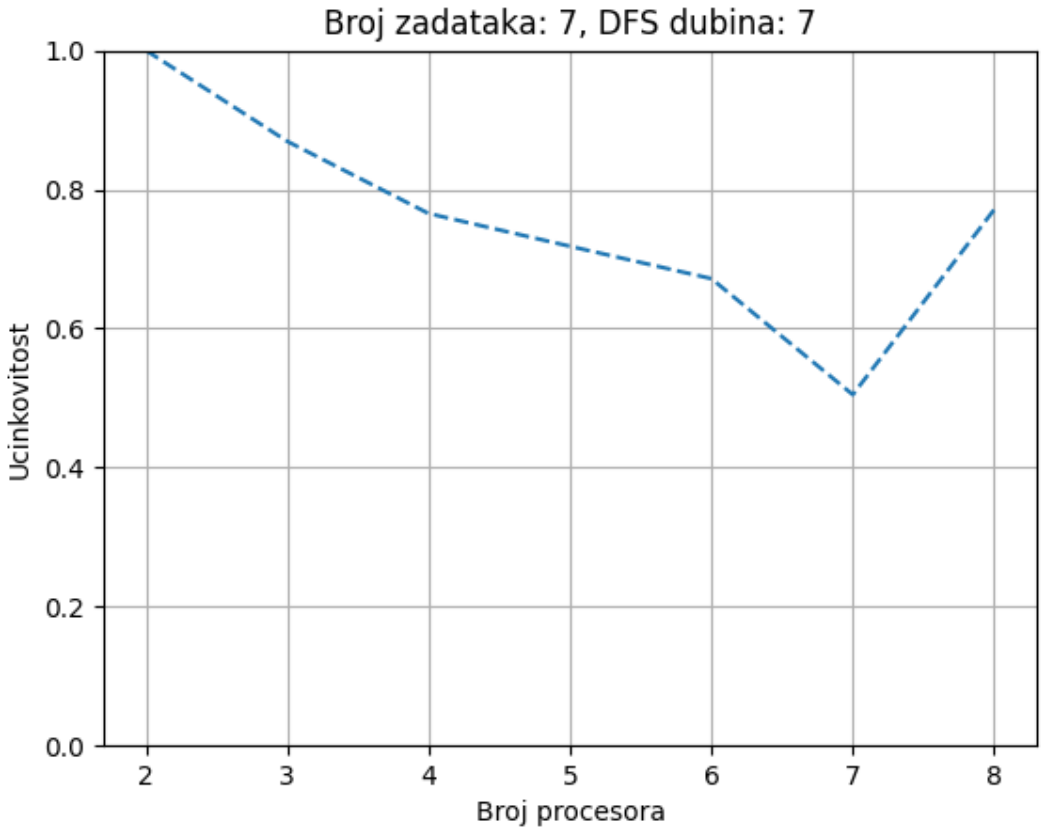
*Upute: U ovom dijelu potrebno je priložiti **tablice s rezultatima mjerenja te grafove ubrzanja i učinkovitosti** za tri različita scenarija: kada paralelni algoritam ima 7, 49 i 343 zadatka (uz aglomeraciju na dubini 1, 2 i 3). Mjerenja treba provesti tako da je najmanje mjereno trajanje (za 8 procesora) reda veličine barem **nekoliko sekundi** (definirajte potrebnu dubinu pretraživanja). Uz grafove, dodajte kratki komentar koji opisuje kako broj zadataka utječe na ubrzanje i učinkovitost (uzevši u obzir utjecaj zrnatosti zadataka, komunikacijskog overhead-a, te udjela programa koji se ne može paralelizirati).*

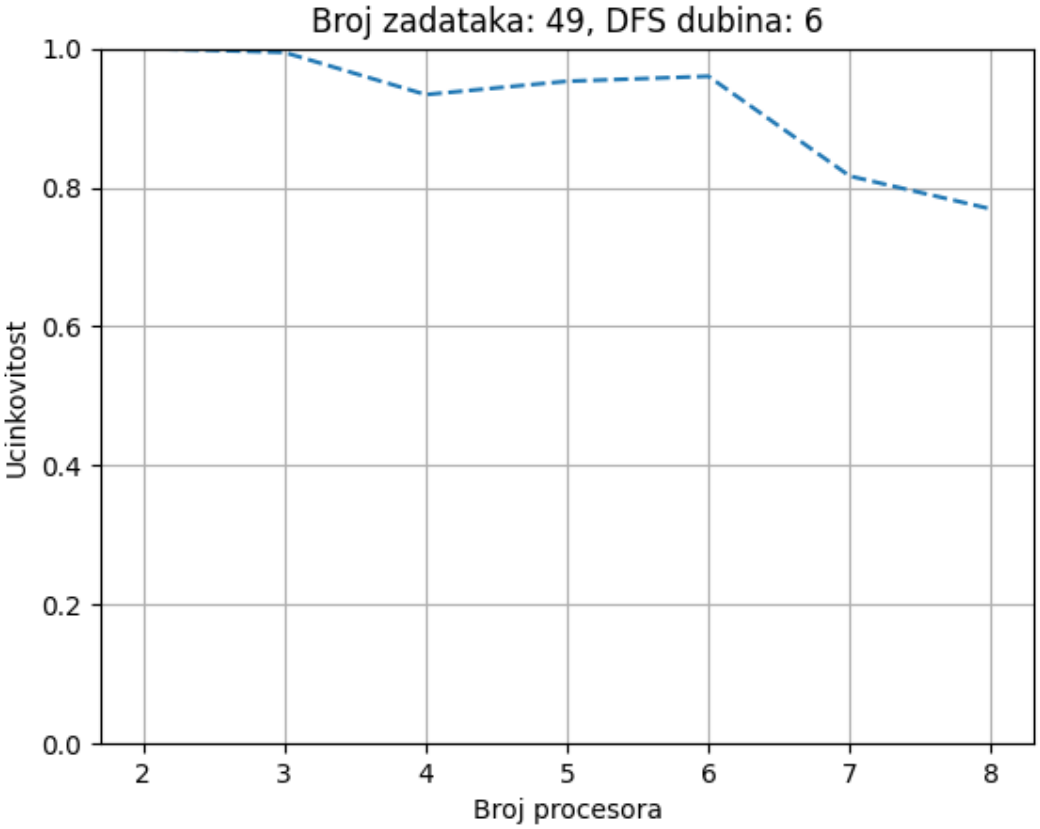
Zad + DFS	**2**	**2**	**3**	**4**	**5**	**6**	**7**
7+7	**18.743**	10.782	8.163	6.524	5.581	6.188	3.473
49+6	**18.522**	9.314	6.613	4.860	3.859	3.781	3.441
343+5	**18.060**	9.415	6.210	4.764	4.193	4.089	3.362

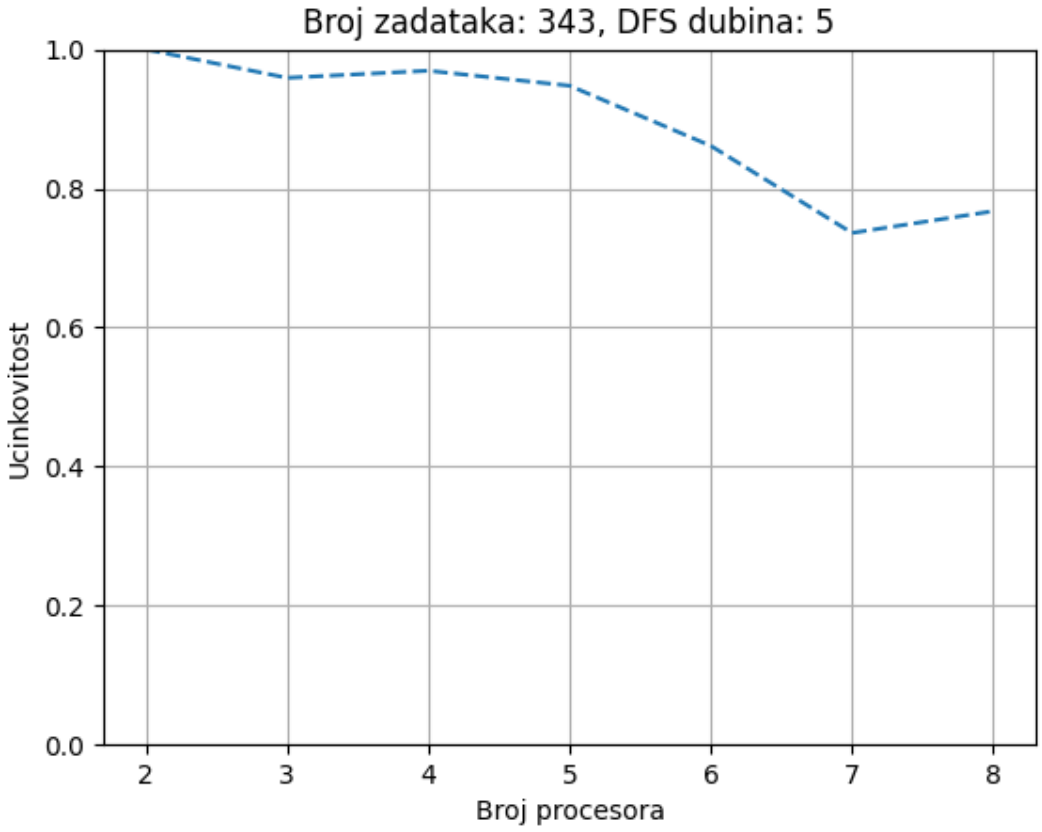

Broj zadataka: 7, DFS dubina: 7
Ubrzanje
Broj procesora
1
2
3
4
5
6
7
8


Broj zadataka: 49, DFS dubina: 6
Ubrzanje
Broj procesora
1
2
3
4
5
6
7
8


Broj zadataka: 343, DFS dubina: 5
Ubrzanje
Broj procesora
1
2
3
4
5
6
7
2
3
4
5
6
7
8


Broj zadataka: 7, DFS dubina: 7
Ucinkovitost
Broj procesora
0.0
0.2
0.4
0.6
0.8
1.0
2
3
4
5
6
7
8


Broj zadataka: 49, DFS dubina: 6
Ucinkovitost
1.0
0.8
0.6
0.4
0.2
0.0
2
3
4
5
6
7
8
Broj procesora


Broj zadataka: 343, DFS dubina: 5
Ucinkovitost
1.0
0.8
0.6
0.4
0.2
0.0
2
3
4
5
6
7
8
Broj procesora

Komentari: