IT081 - Pankhania Aanandi R.

DAA -

Job Assignment Problem using Branch and Bound

Problem Analysis:-

Let there be N workers & N jobs. Any worker can be assigned to perform any job, incurring some cost that may vary depending on the work-job assignment. It is req. to perform all jobs by assigning exactly one worker to each job and exactly one job to each agent in such a way that the total cost of the assignment is minimized.

Example:-

Worker \ Job:	1	2	3	4
A	9	2	7	8
B	6	4	3	7
C	5	8	1	8
D	7	6	9	4

worker A takes 8 units of time to finish job 4.

Approach: It is similar to BFS-like search but with one major optimization. Instead of FIFO order, we choose a live node with least cost. We may not get optimal soln by following node with least promising cost, but it will provide very good chance of getting the search to an ans. node quickly.

Two approaches to calculate cost function:-

1.) for each worker, we choose job with min. cost from list of unassigned jobs [∴ take min. entry from each row.]

2.) for each job, we choose a worker with lowest cost for that job from list of unassigned workers (∴ take min. entry from each col^n)

taking 1st approach, ...

for the given example (above) lets try to calculate promising cost when job 2 assigned to worker A.

	1	2	3	4
A	9	2	7	8
B	6	4	3	7
C	5	8	1	8
D	7	6	9	4

1) ← since job 2 assigned to worker A, job 2 & worker A becomes unavailable.
cost = 2.

	1	2	3	4
A	9	2	7	8
B	6	4	3	7
C	5	8	1	8
D	7	6	9	4

2) ← Now assign job 3 to w-B as it has min. cost from list of unassigned jobs. i.e. {6, 3, 7} = 3
so, job 3 & w-B → unavailable
cost = 2+3 = 5

07 DAY 280-085 SUNDAY

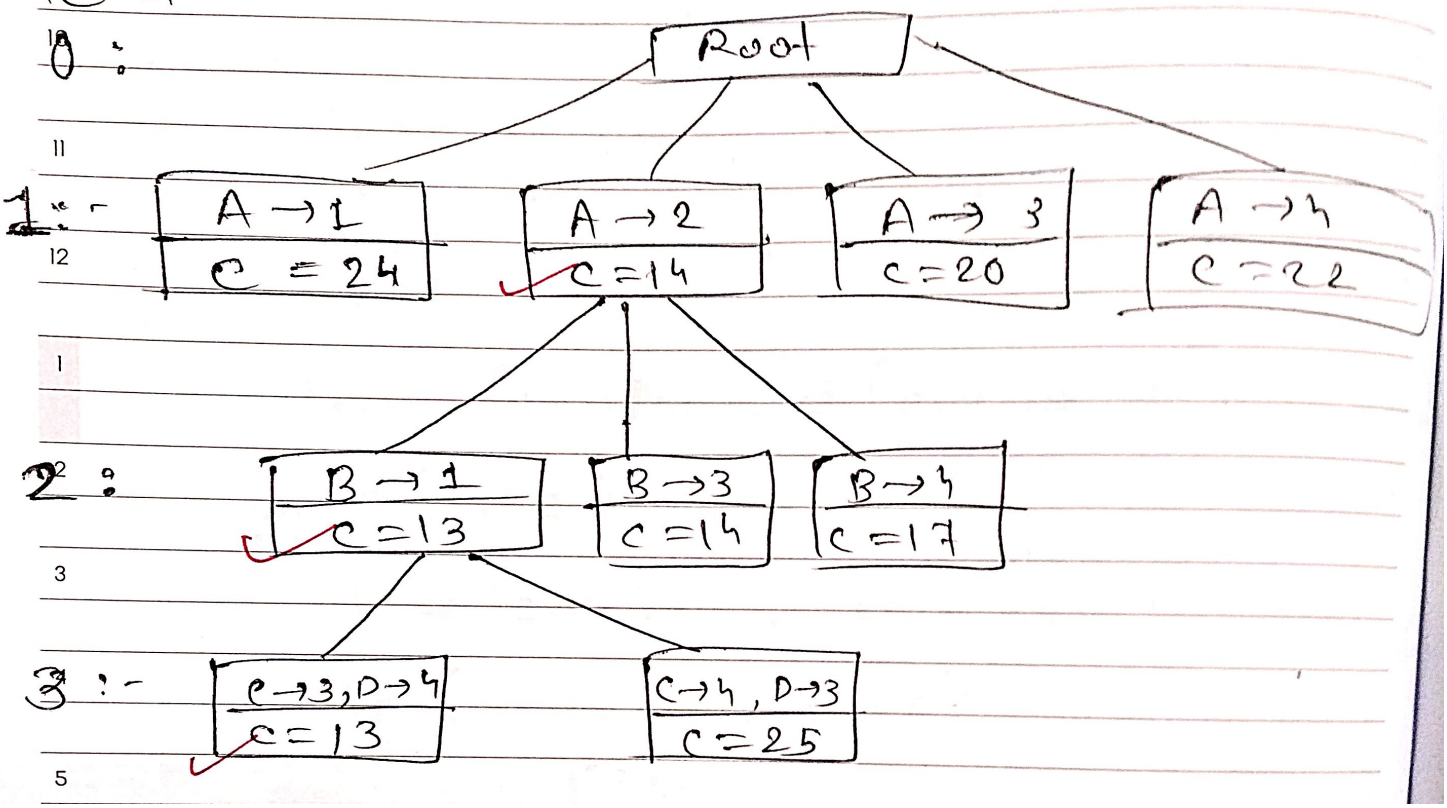
	1	2	3	4
A	9	2	7	8
B	6	4	3	7
C	5	8	1	8
D	7	6	9	4

3) ← then worker C to {5, 8} → job 1 → 5
& cost = 2+3+5 = 10

4) & then finally job 4 to w-D so the total cost becomes, ...
cost = 2+3+5+4 = 14

Search space Tree:-

cost (c)



Algo:-

```
node // i/p:- cost matrix of problem
     // o/p:- optimal cost & Job assignments
node
{
    int job_number;
    int worker_number;
    node parent;
    node cost;
}
algorithm FindMinCost(costMatrix mat[][])
{
    while(true)
    {
        E = least(); // Find node with least cost.
        if (E is leaf node)
        {
            print();
            return;
        }
        for each child x of E
        {
            Add(x); // (add x to list of live nodes.)
            x->parent = E; // pointer for path to root
        }
    }
}
```