***CSCE686 2020  
Homework 2***

***Due (5/12)***

1. *(35pts+)* Select a P-time problem leading itself to a greedy search algorithm*.* Use the given domain/algorithm domain (PD/AD) design process for this P-Time problem by constructing a greedy algorithm from PD through Functional Algorithm using design refinement. (Follow example design as presented in class – see suggested “short and sweet” beginning design outline indicated attached). *Do not use MST (Prim or Kruskal) or SPP.*

*A possible source of problems:* <https://www.hackerearth.com/practice/algorithms/greedy/basics-of-greedy-algorithms/practice-problems/>

***\*\*\*\*\*\*\*\*\*\*\****

***Report outline suggestion for the application of the PD/AD design process per class discussion and notes:***

1. **Title, Introduction**   
   - purpose of design exercise
2. **Define, develop, and analyze** Problem Domain (PD) Specification **(10 points)**  
   - PD requirements – English.  
   - PD symbolic to math/logic formal representation  
    -- input domain and condition; PD data structures(s)  
    -- objective function  
    -- output domain and conditions; PD data structures(s)  
   - class of problem including size of solution space (references)
3. **Select** a Search Algorithm Domain – greedy (AD) based upon problem class **(5 points)**  
   - assume a P-Time problem, (P-Time Complexity, Matroid Class?) use DFS – Greedy to fine optimal solution  
   - use the associated CSCE686 greedy search algorithm template (show)
4. **Evolve** a search Algorithm Design Specification - greedy via integration **(10points)**  
   - integrate problem domain with search algorithm template - greedy  
   - the high-level algorithm design specification  
    -- input domains/output domains and conditions  
    -- integrate PD math/symbolic notation as appropriate  
    -- algorithmic search model as instantiated from search template:  
    a. set of candidates  
    b. next-state generator  
    c. selection function   
    b. feasibility function  
    c. solution function  
    d. objective –fitness function
5. **Expand** Algorithmic Design Specification - greedy into operational form: functional or object-oriented **(10 points)**  
   “ **Refine with numerous design steps”**

“Refine algorithm design recursively to pseudo code level”   
-- the specific functional or OO algorithm selected to design; i.e., transformed from the PD/AD design process  
- input domains/output domains and conditions expanded  
- search specification is further defined and developed:  
 a. set of candidates  
 b. next-state generator  
 c. selection function   
 b. feasibility function  
 c. solution function  
 d. objective –fitness function  
- usually new data structures are defined for more efficient design  
- may involve various design refinement stages (5a, 5b, ..)  
- add comments to each stage of refinement to indicate the current evolution of the design using the search model elements (set of candidates, …) with ability to directly **flow** back design to requirements  
- leads to low-level functional or OO algorithmic design of Pseudo code  
 - usually mapping to specified programming language level   
- determine algorithm **complexity** at functional or OO level

* functional or OO level design should reflect “good” ADT design   
   - axiom definition NOT required but could be addressed  
   - use Talbi’s generic algorithm formulation/presentation
* your design process should explicitly reflect **rational** for development and use of **CREATIVE** data structures and control structures per refinement stage (design process should go slow!)

1. I**mplementation and Test and Analysis (extra credit, 10 points))**-- functional or OO level (pseudo code) mapping to executable code experiment (your choice of language)  
    - Executable code should reflect standard search elements as comments (set of candidates , selection, …)  
    - mapping from pseudo code to programming language code should be close to 1 to 1.  
    - determine objective of your experiments and run (small, medium, and large dimensions)  
    -- analyze time execution as compared to expectation  
    -- Use Barr’s et al suggestions for reporting on computational experiments: (<https://www.dcc.fc.up.pt/~jpp/mpa/Proceedings%20of%20the%20%e2%80%a6%202001%20Barr.pdf>)
2. **Conclusion**, Comments, Recommendations  
   -- address utility of PD/AD design process (effectiveness and efficiency)  
    -- development of specific functional or OO code selected  
    -- discuss possible development of other algorithmic code   
   - discuss impact of PD complexity vs. AD complexity (different!)  
   - briefly discuss variations of the selected problem (references)  
   - other general references and specific application references (limit!)
3. **References** (indicate appropriated references in this design homework)

Notes:

1. Try to Improve on the given top-down design development for better understandability and the proper use of heuristics. More details on   
   PD to AD integration and explicit design steps to functional pseudo code..
2. Use diagrams, figures and tables for conciseness and effect.
3. Could use Word or Latex manuscript software allowing more formalized reporting: table of contents, list of figures, index, …
4. One could do a bottom-up design first since the greedy algorithm implementation code or pseudo code maybe found. Nevertheless, the HW requires a top-design that stands on its own ( A new reader would after reading your top-down design, would know exactly all the explicit design decisions leading to the greedy pseudo code.).