Gedd Johnson

Project 4 Bin Packing

Data Structures and Algorithms II

**Functional Decomposition for Project 4**

|  |  |  |
| --- | --- | --- |
| File | Functions | Purpose |
| Item.h | None | Public implementation of Item ADT. |
| Item.c | 1.) ItemP createItem(int);  2.) ItemP freeItem(ItemP); | 1.) Creates a new Item ADT. This is basically a node.  2.) Frees the memory used by an Item ADT. |
| Bin.h | None | Public implementation of Bin ADT. |
| Bin.c | 1.) ListP createBinList();  2.) BinP addBinToList(ListP, int);  3.) ListP resetBinList(ListP);  4.) void printBins(ListP);  5.) ListP freeBinList(ListP); | 1.) Creates a BinList ADT.  2.) Creates and adds a Bin ADT to the BinList.  3.) Frees and clears the all Bins and Items in a BinList then creates a new BinList in preparation for the next run.  4.) Prints the contents of the BinList including all Bins and Items.  5.) Frees all memory used by the Binlist including all Bins and Items. |
| Greedy.h | None | Public implementation of all greedy algorithms including first, next, and best fit. |
| Greedy.c | 1.) void OnlineFirstFit(ListP, int\*, int);  2.) void OnlineNextFit(ListP, int\*, int);  3.) void OnlineBestFit(ListP, int\*, int);  4.) void OfflineFirstFit(ListP, int\*, int);  5.) void OfflineBestFit(ListP, int\*, int);  6.) void FirstFit(ListP, ItemP);  7.) BinP NextFit(ListP, ItemP, BinP);  8.) void BestFit(ListP, ItemP);  9.) void placeItemInBin(BinP, ItemP);  10.) BinP itemTooLarge(ListP, BinP, ItemP);  11.) void mergeSort(int\*, int);  12.) void mergeBack(int\*, int\*, int , int\*, int); | 1.) Performs the Online First Fit algorithm.  2.) Performs the Online Next Fit algorithm.  3.) Performs the Online Best Fit algorithm.  4.) Performs the Offline First Fit algorithm.  5.) Performs the Offline Best Fit algorithm.  6.) Performs the First Fit algorithm regardless of on or off line.  7.) Performs the Next Fit algorithm regardless of on or off line.  8.) Performs the Best Fit algorithm regardless of on or off line.  9.) Places an Item into the current Bin and updates the current size of that Bin.  10.) If an Item is too large for all the Bins in the BinList, this function creates a new Bin and either places the Item in the Bin or throws it away.  11.) Merge sort algorithm used for sorting the Items in descending order by size.  12.) Helper for the merge sort function, merges array back together in descending order. |
| main.c | 1.) int main(void);  2.) void printTables(ListP, ListP, ListP, ListP, ListP); | 1.) Solves the bin packing problem using all 5 greedy algorithms in both online and offline versions and prints results to the screen.  2.) Prints results of each algorithm after each run including a table and contents of each bin. |

Data Structures:

struct Item{

int size;

ItemP nextItem;

};

struct BinList{

int numBins;

BinP head;

};

struct Bin{

int capacity;

int currentSize;

BinP nextBin;

ItemP firstItem;

};