# Generalizing Slotting Strategy

**Goal:** Given a csv file representing a grid of any size containing pick frequency data (like in LAB 12) output a completed and optimized new grid containing the SKU locations.

## Problem assumptions:

1. Must take a warehouse grid of frequency data of any size (rows,aisles) as a CSV file
2. **Dock location**
3. **Obstacle locations (if any)**
4. Unique SKUs are at each position (row,aisle)
5. SKU0 is located at (0,0), SKU1 → (0,1); row-wise across the grid
6. A = top 20% B = 30% C = 50% (allow for user defined %s)

## Problem functions:

1. Load frequency grid.
2. Flatten the grid to skus.
3. Conduct paretto
4. Calculate abc classes.
5. Assign skus to slots.
   1. K

## Problem output (goal):

### **LAB 12: Generalizing Slotting Strategy Tool – Problem Description**

Objective:

Design a tool that automates the process of slotting inventory items (SKUs) in a warehouse by optimizing placement based on pick frequency and proximity to the dock, while generalizing the method to work on any warehouse grid size.

### **Detailed Problem Description**

We are given a CSV file that represents a warehouse grid of any size (e.g., 5x5, 10x3, etc.), where each cell contains the pick frequency for a specific SKU located at that grid position. The overall goal is to reorganize the SKUs in the warehouse such that high-frequency SKUs (A class) are closest to the dock, followed by medium (B) and low (C) frequency SKUs, considering any obstacles and allowing user-defined ABC percentages.

### **Inputs**

1. Warehouse Frequency Grid (CSV file):  
     
    A 2D grid with pick frequencies, where each cell corresponds to one SKU. The SKU at (0,0) is SKU0, and SKUs continue row-wise:  
     
    SKU0 = (0,0), SKU1 = (0,1), ..., SKUn = (r, c)
2. Dock Location:  
     
    One or more coordinates representing where the dock(s) are located in the grid. Proximity to the dock influences slotting priority.
3. Obstacle Locations (optional):  
     
    List of coordinate pairs that are blocked/unavailable and must be skipped during slotting.
4. ABC Percentages (customizable):  
     
    Default percentages are A = top 20%, B = next 30%, C = remaining 50%, but the user should be able to modify these.

### **Functional Steps**

1. Load CSV:  
     
    Read and store the frequency data from the input CSV into a structured format (e.g., 2D list or NumPy array).
2. Flatten Grid:  
     
    Convert the 2D frequency grid into a list of (SKU, frequency, original location) tuples.
3. Sort by Frequency:  
     
    Apply a Pareto analysis to rank SKUs by frequency in descending order.
4. Assign ABC Classes:  
     
    Classify SKUs into A, B, or C based on cumulative percentages of total frequency or item count using the specified or default thresholds.
5. Calculate Distance to Dock:  
     
    For each valid (non-obstacle) grid cell, calculate its distance to the nearest dock point using Manhattan or Euclidean distance.
6. Generate Slot Priorities:  
     
    Create a ranked list of available (non-obstacle) grid cells, sorted by proximity to the dock.
7. Assign SKUs to Slots:  
     
    Allocate the highest-frequency SKUs (A class) to the closest available cells, then B, then C, filling the grid accordingly.

### **Output**

* Optimized Warehouse Grid:  
    
   A new CSV/grid structure that displays the reassigned SKU locations in the warehouse based on optimized slotting.
* Visual Summary (optional):  
    
   A visualization or heatmap showing the new slotting layout and distances from the dock.

### **Prompt Version (Copy-Paste to Build the Tool):**

Design a generalizing warehouse slotting optimization tool that takes a CSV representing any grid size (rows x columns) of SKU pick frequencies. Each SKU is initially located at a grid cell starting from SKU0 at (0,0), continuing row-wise. The tool must: 1) Load the CSV, 2) Flatten the grid into SKU-frequency-location tuples, 3) Apply Pareto sorting, 4) Assign SKUs to ABC classes using customizable percentage thresholds (default A=20%, B=30%, C=50%), 5) Accept user input for dock and obstacle coordinates, 6) Rank all available (non-obstacle) slots by proximity to the dock, and 7) Reassign SKUs to these slots with A items closest to the dock, followed by B and C. Output the new optimized SKU grid as a CSV.